VŠB– Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Síťový kamerový systém s datovým přenosem obrazového signálu

Network Camera System with Data Transfer of Image Signal

2018/2019 Tomáš Hudák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Tomáš Hudák**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2612R041 Řídicí a informační systémy

Téma: Síťový kamerový systém s datovým přenosem obrazového signálu
Network Camera System with Data Transfer of Image Signal

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Rozbor problematiky základních principů a způsobů datových přenosů a jejich ochrana.
2. Rozbor problematiky zpracování obrazového signálu.
3. Návrh síťového kamerového systému.
4. Realizace síťového kamerového systému.
5. Verifikace a optimalizace síťového kamerového systému.
6. Srovnání naměřených výsledků s teoretickými předpoklady.
7. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] OZER, Jan. *Producing Streaming Video for Multiple Screen Delivery.* Doceo Publishing, 2013. ISBN 978-0976259541.
[2] AUSTERBERRY, David. *The Technology of Video and Audio Streaming.* 2 edition. Focal Press, 2004. ISBN 978-0240805801.
[3] GHANBARI, Mohamed. *Standard Codecs: Image Compression to Advanced Video Coding.* London,UK: The Institution of Engineering and Technology, 2003. ISBN 978-0852967102.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeněk Macháček, Ph.D.**

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2019

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Čestné prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 30. 4. 2019

. . . . . .
Podpis studenta

Poděkování

Děkuji panu Zdeňkovi Macháčkovi za odborné vedení práce při konzultování a vypracovávání.

Abstrakt

Cílem této práce je vytvořit kamerový systém umožňující provozovat video hovory pomocí sítě na standartu UDP. Tento kamerový systém je schopen provozovat strukturované hovory paralelně mezi mnoha uživateli zároveň, obdobně jako skupinová videokonference nebo VoIP jako Skype. Dále lze tento systém využít pro monitorování zabezpečených prostorů.

Vhodná volba kompresního videoformátu umožňuje přenášet proud videa v dobré kvalitě i za nižšího datového toku což umožňuje přenášet obraz i při pomalém připojení v přijatelné kvalitě. Celý systém má být dynamický což znamená že se musí umět přizpůsobit nově připojeným klientům a zobrazit jejich dostupné kamery.

Systém je založen na embeded zařízení Raspberry PI 3, které má malé rozměry a dostatečný výkon s GPU na přehrávání videa a vhodný počet USB portů na zapojení maximálně čtyř USB kamer.

Abstract

The aim of this work is to create a camera system allowing video calls over UDP protocol. This camera system has the ability to communicate with many users at the same time, like group video conferencing or VoIP like Skype. In addition, this system can be used to monitor secure areas.

Appropriate video-format choices allow you to stream video streams in good quality even at a lower bit rate, allowing you to transfer images even with slow connections at acceptable quality. The entire system should be dynamic, which means that it must be able to adapt to newly connected clients and display their available cameras.

The system runs on embedded devices Raspberry PI 3, which has small dimensions and enough power with the video playback GPU and a suitable number of USB ports to connect at least four USB cameras.

Obsah

Seznam použitých zkratek a pojmů

Zkratka	Význam
HW	Hardware
ISO	International Organization for Standardization
RS-232	Standart komunikačního rozhraní
10Base2	Základní standart sítě Ethernet (10 Mbit/s)
OSI	Open Systems Interconnection
Skype	Software pro vzdálenou komunikaci umožnující video hovory
OpenCV	Knihovna pro manipulaci s obrazem
WIFI	Komunikační standart pro bezdrátový přenos dat
USB	Universal serial bus
DHCP	Dynamic host configuration protocol
SW	Software
RPi	Raspberry
MPEG	Moving Picture Experts Group
FFMPEG	Program pro konverzi audio a video formátů
CPU	Central processing unit (procesor)
GPU	Graphics processing unit (grafický procesor)
FPS	Frames per second – počet snímků za sekundu
IP	Internet Protocol
MAC	Media Access Control
CAN	Sériová sběrnice (Controller area network)
I^2C	Sériová sběrnice (Inter-Integrated Circuit)
GNU	(GNU is Not Unix) projekt zaměřený na svobodný software
CCD	Charge coupled device
GOP	Group of pictures
VBR	Variable Bitrate
DCT	Discrete cosine transform
HDR	High dynamic range
INT	Celé číslo (integer)
FLOAT	Desetinné číslo
ms	milisekunda

Seznam obrázků

Seznam tabulek

1. Úvod

Již delší dobu se můžeme setkat s živým vysíláním videa po síti. Můžeme například sledovat přímý přenos televize přes internet, nebo můžeme přes síť monitorovat prostor pomocí CCTV kamery. Tyto kamery můžeme vidět na různých zabezpečených místech jako jsou parkoviště, banky či jiné zabezpečené prostory. Existují sítě, které využívají více kamer. Tyto kamery lze sledovat současně naráz a můžeme je zabezpečit vůči neoprávněným uživatelům oddělením od veřejné sítě a nechat je oddělené v intranetu.

Tato práce se zaměřuje na přenos multimediálního vysílání po síti. Pomocí programu FFMPEG je zachycen obraz z web kamery, který je pak zkomprimován na mešní velikost při níž se zmenší datový tok. Dále je pak obraz vysílán v zkomprimované podobě po síti prostřednictvím skupinových adres. Uživatel si může vybrat které kamery chce zveřejnit v síti prostřednictvím grafického rozhraní, které je napsané v C++ s využitím frameworku Qt.

První část práce rozebírá základní principy přenosu dat v sítích. Jsou zde popsány různé metody přenosu dat, typy signálů, které se využívají k přenosu. Následně jsou zde popsány různé síťové vrstvy a jejich protokoly. Dále jsou zde popsány způsoby přeposílání datových rámců v síti, s využitím multicast IP nebo broadcast IP adres, které umožňují směřovat vysílání více pozorovatelům najednou. Jsou zde také popsány způsoby kontroly dat a popsány jejich metody. V poslední části kapitoly jsou popsány kompresní metody videí.

Další část této práce se zabývá návrhem tohoto systému. Zde je popsán vzhled grafického rozhraní a jednotlivé principy, které jsou použity k realizaci tohoto systému. Jsou zde popsány nároky na výkon kódování a přenos dat po síti a popis komunikace mezi serverem a klientem. Je zde popsán vybraný formát kódování. Je zde také popsán způsob zapojení jednotek do sítě do centrálního síťového prvku jako je switch.

Následující kapitola popisuje realizaci síťového kamerového systému. Je zde popsán vybraný hardware a jeho specifikace. Je zde také popsáno zapojení jednotek Raspberry do sítě prostřednictvím Ethernetu nebo WIFI. V další části je jsou zde popsány části kódu v C++ s použitím knihovny OpenCV. Dále jsou zde popisy vlastních protokolů pro komunikaci mezi serverem a klientem Nacházejí se zde také zjednodušené vývojové diagramy jejich cílem je naznačit jakým způsobem daný algoritmus pracuje.

2. Rozbor problematiky základních principů a způsobů datových přenosů a jejich ochrana.

Datový přenos je druh komunikace umožňující transportovat data v elektronické formě. Přenos se skládá z dvou nebo i více bodů, kde hlavní roli hraje přenosové médium jako je metalický kabel nebo optický kabel či rádiový přenos.

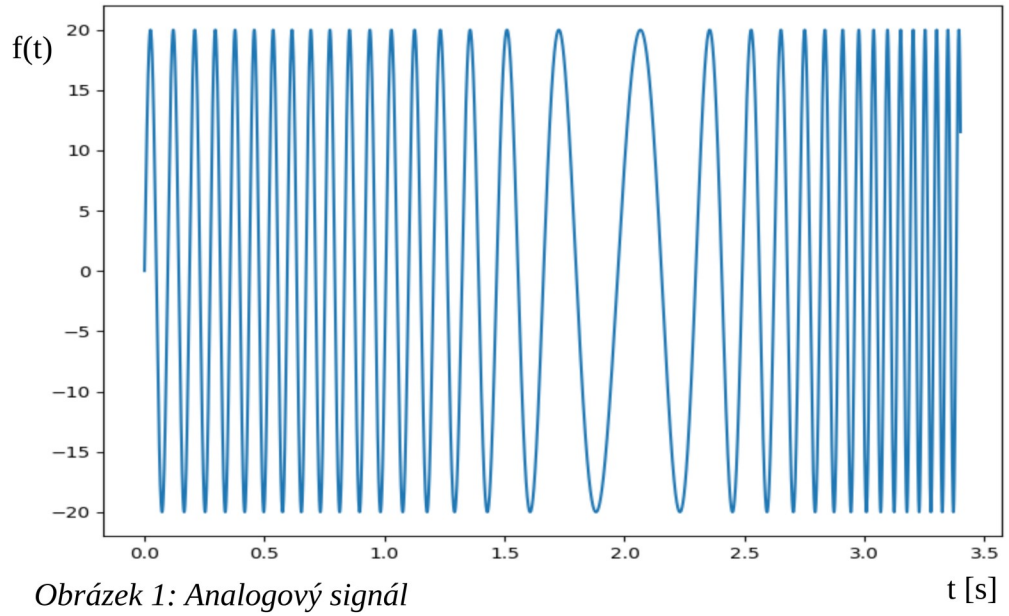
Komunikace probíhá na obou stranách, kdy na jedné straně jeden bod vysílá a na druhé straně protější bod přijímá. Přenos se tedy skládá ze dvou činností, vysílání a příjímání.

2.1 Způsoby datového přenosu

Data mohou být přenášená ve dvou formách. Nejzákladnější forma datového přenosu je analogová, jejíž signál je spojitý v čase a může nabývat neomezené škály hodnot. Druhou formou přenášení dat je pomocí digitálního signálu. Ten je v čase diskrétní a má pevně danou vzorkovací periodu.

2.1.1 Analogový přenos

Tento přenos využívá signálu, který je v čase spojitý a jeho okamžité hodnoty nejsou omezené přesně daným rozpětím. Tento signál se používá například k přenosu zvuku po kabelu nebo rádiové vysílání rozhlasu. Přirozeným analogovým signálem je sluneční světlo.

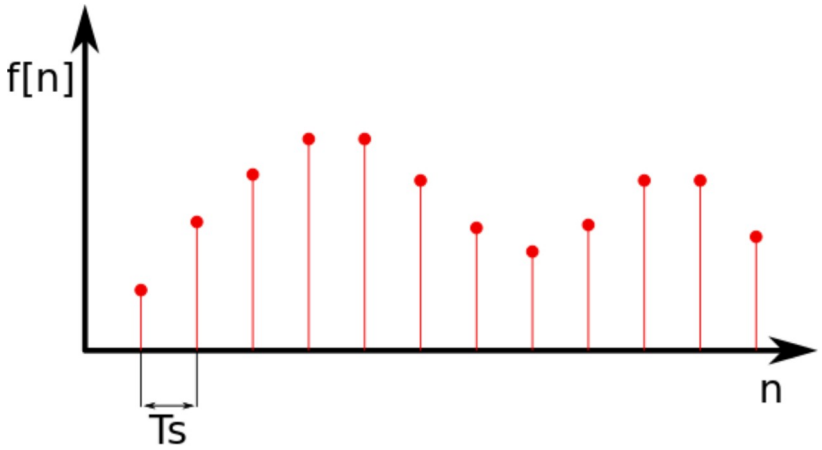


Obrázek 1: Analogový signál

2.1.2 Přenos diskrétním signálem

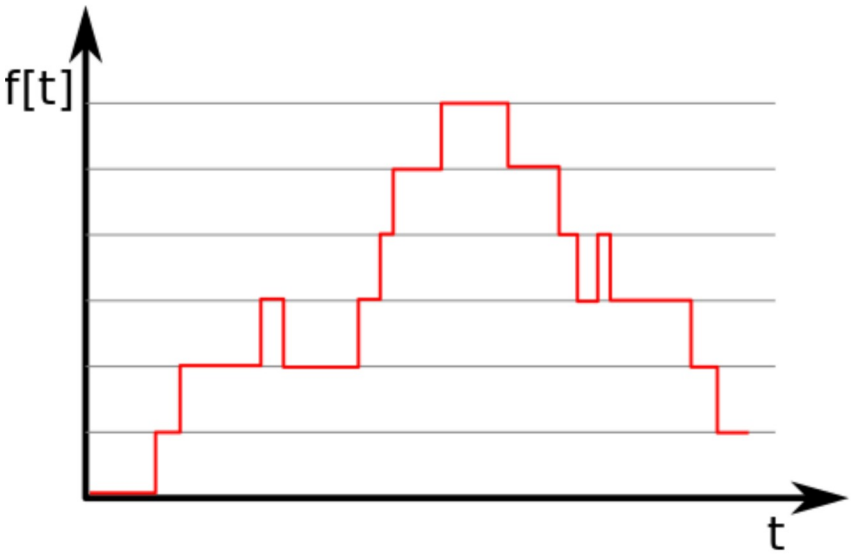
Tento druh přenosu využívá signál, který je v čase diskrétní což znamená že má hodnoty definované jen v určitých časových intervalech, které jsou pevně dané vzorkovací periodou. Jednotlivé úseky v každé vzorkovací periodě se nazývají vzorky.

Vzorkovaný signál
Tento druh signálu má pevné daně rozpětí hodnot na časové ose. Rozpětí je dané vzorkovací periodou, která je konstantní a její obrácená hodnota udává vzorkovací frekvenci, která udává počet vzorků za sekundu.



Obrázek 2: Vzorkovaný signál

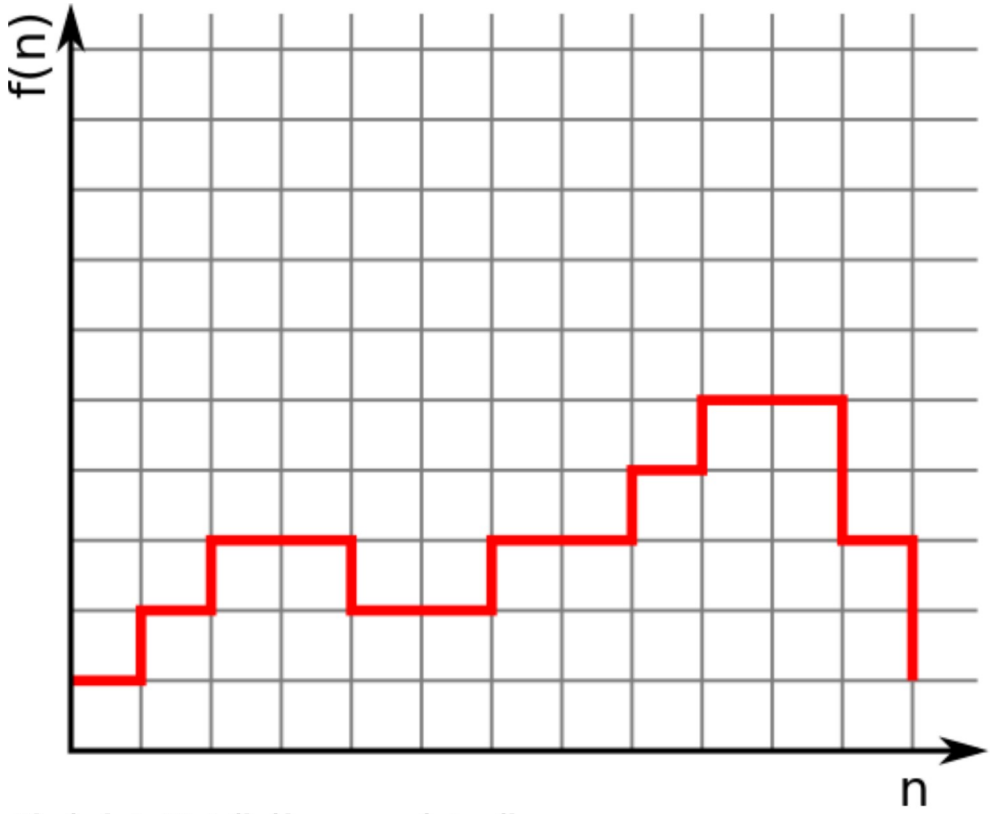
Kvantovaný signál
Tento signál nabývá pouze určitých hodnot, dané omezeným počtem úrovní s konstantním rozpětím. Signál je v čase spojitý a změna hodnoty může nastat v libovolném čase, v amplitudě je však diskrétní .



Obrázek 3: Průběh kvantovaného signálu

Tento druh signálu může být například výstupem paralelního D/A převodníku.

Digitální signál
Hodnoty digitálního signálu jsou dány jeho bitovou hloubkou která určuje rozlišení kvantování, nemůže tedy nabývat libovolných hodnot tak jako je u analogového signálu. Rozlišení hodnot je závislé na počtu bitů, které tvoří jednu hodnotu v signále. Signál může být například osmibitový (nabývá pouze 256 hodnot) nebo šestnáctibitový (může nabývat 65536 hodnot).



Obrázek 4: Digitální kvantovaný signál

K tomu aby byla přenesena hodnota digitálního signálu je třeba tento signál přenášet po bitech, které mají danou přenosovou rychlost (bitrate nebo též baudrate). Bity lze přenášet sériově nebo paralelně.

2.2 Vrstvy digitálního přenosu dat

Vrstvy digitálního přenosu dat se dělí podle modelu ISO/OSI. Model ISO/OSI se snaží standardizovat telekomunikaci mezi výpočetními systémy. Při tom využívají určitých norem, které uvádění všeobecné principy, které jsou rozčleněny do sedmi vrstev. Tento model popisuje funkce každé vrstvy.

V ISO/OSI modelu je celkem 7 vrstev.

- **Fyzická vrstva –** Zajišťuje přenos informací skrze transportní médium (drát, rádiové vlny, optika). Definuje způsob přenášení bitů, nastavení logických hodnot dle úrovně napětí.

- **Linková vrstva –** Zajišťuje přenos dat mezi jednotlivými síťovými uzly a rozlišuje je podle adres (MAC). Je schopná také detekovat chyby v přenosu.

- **Síťová vrstva -** Stará se o směrování v síti a síťové adresování a poskytuje spojení mezi zařízeními, které se nacházejí na odlišných místech. Protokol, který pracuje na této vrstvě se nazývá Internet Protokol.

- **Transportní vrstva –** Stará se o adresaci aplikací v systému pomocí čísel portů v systému. Poskytuje spolehlivý přenos dat.

- **Relační vrstva –** Obstarává způsob otevírání a zavírání relací mezi aplikacemi pro koncové uživatele. V případě dočasné ztráty spojení se protokol této vrstvy se snaží toto spojení obnovit.

- **Prezentační vrstva –** Úkolem této vrstvy je formátovat informace určené pro aplikační vrstvu. Vrstva se zabývá převodem struktury dat, kterou převádí na

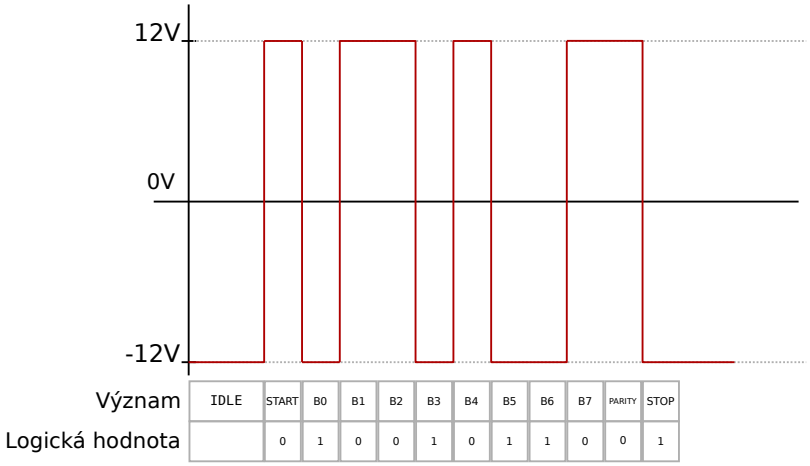
- **Aplikační vrstva –** Vrstva umožňující aplikacím a komunikačním systémům mezi sebou pracovat.

2.2.1 Fyzická vrstva

Tato vrstva zajišťuje převod proudu bitů na signál (nejčastěji elektrický) a opačně. Tato vrstva má protokoly, které rozhodují o napěťových úrovních přenosových rychlostech a modulacích.

Existuje několik standardů pracujících na fyzické vrstvě.

- **RS-232 –** Tento standard slouží jako komunikační rozhraní jehož účelem je propojit dvě zařízení, které spolu můžou provozovat sériovou komunikaci (tzn. Jednotlivé bity jsou přenášeny za sebou).



Obrázek 5: Znázornění komunikace RS232

- **10Base2 –** Standart sítě Ethernet založený na koaxiálním kabelu.

- **1-Wire –** Tento standard umožňuje přenášet data sériově po dvou drátech, kde jeden drát je zem a druhý je datový. V tomto standardu je třeba mít jedno hlavní zařízení (Master) a zbylá zařízení (kterých může být i více) se poté chovají jako Slave.

- **CAN Bus –** Jedná se o sériovou sběrnici umožňující komunikaci mezi senzorama uvnitř automobilu.

- **I^2C –** Tato sběrnice využívá celkem 3 vodiče, kde jeden udává časování (SCL), druhý posílá data obousměrně (SDA), a třetí je napájecí. Hodnota SDA se mění pouze je li SCL na nízké hodnotě (LOW)

2.2.2 Linková vrstva

Tato vrstva zajišťuje komunikaci mezi dvěma nebo více uzly, které jsou rozlišené na základě hardwarové adrese (MAC). Zajišťuje logiku řízení přenosu a shlukuje data do paketů. Tato vrstva používá protokoly, které jsou schopny detekovat chyby a opravit je.

Mezi standarty na této vrstvě patří:

- **Ethernet –** Protokol využívající přenosu dat po kabelu, většinou kroucenou dvojlinkou se čtyřmi páry (osm vodičů).

- **Wi-Fi –** Označení pro standart bezdrátového vysokorychlostního přenosu v bezlicenčním rádiovém pásmu.

- **Bluetooth –** Tato technologie spadá do sítě osobních počítačů (PAN – personal area network). Jedná se o bezdrátovou technologii pracující na frekvencích 2,402 GHz až 2,480 GHz.

- **PPP –** Tento protokol (zkratka Point to point protocol) je používán pro příjmé spojení mezi dvěma síťovými uzly. Umožňuje autentizaci a šifrování, kompresi, detekci chyb.

2.2.3 Síťová vrstva

Tato vrstva je zodpovědná za směrování a hledání nejvhodnější cesty paketů. Obstarává síťové adresy, podle níž se můžou zařízení v síti spojit a komunikovat.

Adresa jednoznačně rozlišuje jednotlivě zařízení v síti.

- **IPv4 –** Tento druh adresování je dvaceti čtyř bitový, adresa je rozdělaná na 4 části po osmi bitech. Adresa může mít například tuhle podobu: 192.168.1.112. Celkem může být 4 294 967 296 adres v síti.

- **IPv6 –** Tento druh adresování je podobný výše zmíněnému, ale s rozdílem že přináší mnohem větší adresní prostor (celkem 2^{128} adres**).**

Adresy lze nastavovat ručně nebo automaticky. K automatickému nastavování těchto adres slouží DHCP server.

DHCP server (Dynamic Host Configuration Protocol)

DHCP server umožňuje připojeným klientům v síti automaticky nastavit IP adresy. Tím zjednodušuje správu sítě, není tedy nutnost aby každé zařízení v síti si manuálně nastavilo statickou IP adresu.

Klient, který zpočátku nemá adresu chce zažádat DHCP klienta o adresu. Tento klient vyšle broadcast zprávu že žádá DHCP server adresu a zpětně mu DHCP server odpoví s nabídkou IP adresy. Tato IP adresa je časově omezená, neboť DHCP server určuje čas zapůjčení (*lease time*).

V sítích používající IPv4 adresy se používají celkem 3 typy adres:

- **Adresa zařízení (Inet address).** Tato adresa je konkrétní pro jednotlivá zařízení, každé zařízení má svou jedinečnou adresu. Adresa může být například 192.168.1.113.

- **Maska sítě (Netmask).** Určuje rozsah adres v dané podsíti. Tento rozsah je dán podle formy masky, kde je číslo 255 tam se čísla IP adres nemění. Pokud je číslo v masce jiné než 255, například 0 pak je v IP adrese na témže místě celý rozsah 0 – 255 .Pokud tedy maska je

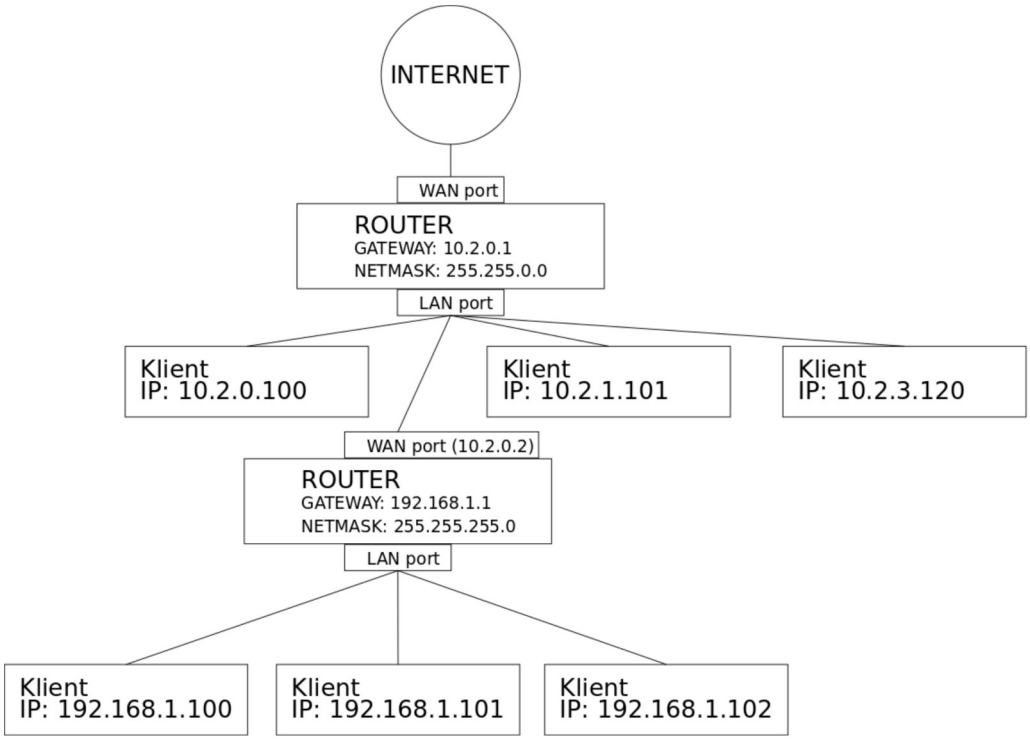
255.255.255.0 znamená to že se můžou měnit čísla pouze na konci. V tomto případě bude rozsah možných adres 192.168.1.0 - 192.168.1.255.

- **Adresa brány (Gateway).** Tato adresa patří hlavnímu uzlu v síti který se stará o směrování mezi vnitřní síti (LAN) a vnější síti (WAN). Zařízení se na tuto adresu vždy připojují a vysílají na ní požadavky pokud se chtějí dostat mimo místní síť.

Tabulka 1: Rozsahy adres podle masky

Maska	**IP adresy**
255.255.255.0	192.168.1.0 – 192.168.1.255 (celkem 256 možností)
255.255.0.0	192.168.0.0 – 192.168.255.255 (celkem 65536 možností)

Příklad:



Obrázek 6: Příklad sítě s více rozsahy

Na obrázku 6 můžeme vidět síť s více rozsahy, které jsou rozděleny routery. Tyto rozsahy mají rozdílnou síťovou masku a oba jsou rozlišeny podle sítí (10.2.0.0/16 a 192.168.1.0/24). Každý síťový uzel náležící do určité sítě musí mít stejnou formu adresy například uzel patřící do sítě 192.168.1.0/24 může mít adresu začínající pouze na 192.168.1.x .

2.2.4 Transportní vrstva

Tato vrstva poskytuje protokoly zajišťující přenos dat mezi koncovými uzly. Tyto protokoly se dělí podle toho jestli zajišťují záruku na doručení dat.

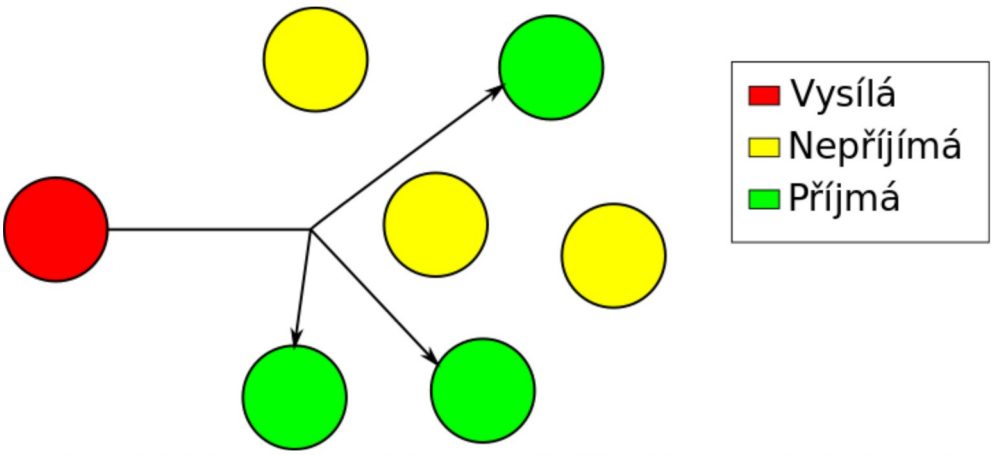
- **Protokol UDP –** Neboli User Datagram Protokol je druh protokolu, který se nestará o správnost doručení datagramu, nemá tedy zpětnou vazbu, zda klient paket přijal. Pokud tedy dojde k nějaká ztrátě datagramu nebude se tento protokol znovu zdržovat jeho opětovným zasláním. Tento protokol je tudíž vhodný pro přenos videa nebo přenosu zvuku po síťi.

- **TCP – Transmission Control Protocol –** Tento protokol na rozdíl od UDP garantuje spolehlivé doručování datagramů. TCP na straně příjemce posílá zpět potvrzení pro pakety, které byly úspěšně přijaty. Služby které používají tento protokol jsou například webový server, mail nebo databáze.

2.3 Způsoby přeposílání datových rámců v síti

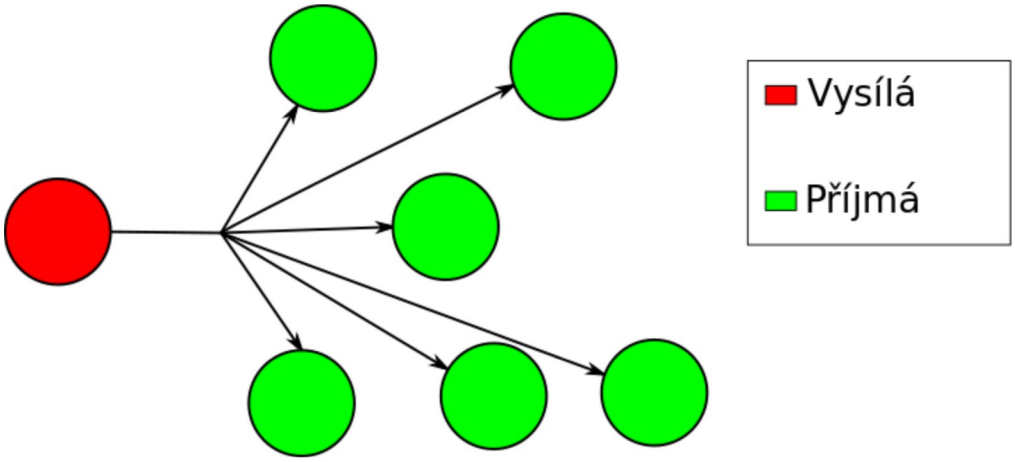
Datový rámec je tvořen množinou Bytů, které jsou posílány v síti mezi dvěma nebo více komunikačními uzly. Existuje několik metod přeposílání datových rámců po síti. Tyto metody se dělí podle způsobů předávání paketů jednotlivým přijímačům. Obvykle vysílacím síťovým prvkem bývá server a přijímacím prvkem je klient. V této práci jsou použity jen dva typy:

- **Multicast –** Tento styl komunikace je nejvhodnější pro vysílací účely, kde jeden vysílací síťová uzel vysílá paket, který je určen všem, kteří se jej rozhodnout přijímat. Vysílací uzel nemusí každému přijímacímu uzlu posílat paket zvlášť. Díky tomuto modelu přenosu dat lze vysílat mnoha klientům naráz, kteří ve stejný čas obdrží totožná data. S rostoucím počtem přijímačů (klientů) se zatížení vysílacího serveru nijak nezvyšuje, protože se server nemusí starat o každý přijímač zvlášť (obslouží všechny najednou). Pro identifikaci se používají multicastové adresy v rozsahu 224.0.0.0 – 239.255.255.255.



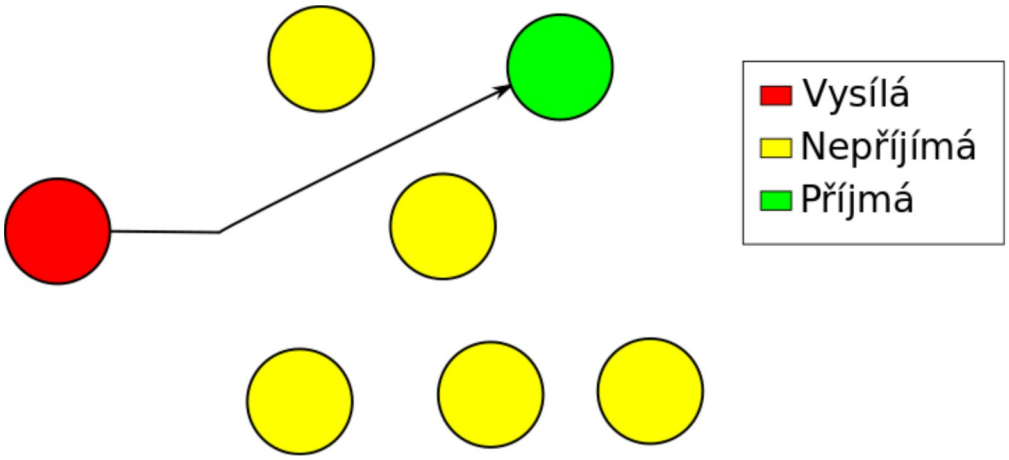
Obrázek 7: Schéma směrování dat mezi několika síťovými uzly (Multicast)

- **Broadcast –** Tento druh komunikace je podobný Multicastu, s tím rozdílem že data pošlou všem v síti. Pro tento druh komunikace se používá pouze jedna adresa spadající do síťového rozsahu, například 192.168.1.255.



Obrázek 8: Schéma směrování dat mezi síťovými uzly (Broadcast)

- **Unicast –** Druh komunikace pouze jeden s jedním. Zde je jeden vysílací uzel a pouze jeden uzel přijímací. Server musí vyslat pakety každému klientovi zvlášť, pokud je klientů více, server může být vytížen, což se projeví negativně na rychlosti. Příkladem může být více klientů, kteří stahují soubory z jednoho serveru, rychlost se mezi ně dělí a s rostoucím počtem klientů rychlost stahování pro jednotlivé klienty se snižuje, což má za následek zvýšení odezvy serveru.



Obrázek 9: Schéma směrování dat mezi dvěma síťovými uzly (Unicast)

2.4 Popis komunikace mezi klientem a serverem

Komunikace mezi serverem a klientem je popsána vztahem kdy na jedné straně se nachází klientská aplikace, který žádá o služby vzdálený program zvaný server, který je poskytovatelem těchto služeb.

Klient je aktivní síťová komunikační strana, která posílá požadavky na server, čeká a dostává odpovědi ze strany serveru.

Server je pasivní strana, která má otevřený port na němž naslouchá a je připraven přijmout žádosti klientů. Při přijetí žádosti je obslouží. Může posílat data, které si klient vyžádá. Při velkém počtu klientů, kteří vysílají požadavky ve stejném čase najednou, je server přetěžován.

2.5 Šifrování datového přenosu

Šifrování je proces při kterém se data pomocí šifrovacích algoritmů převádí na data šifrovaná, tudíž nečitelná běžnému uživateli, který nemá k dispozici klíč k dešifrování. Šifrování je způsob jakým se dá zamezit nepovoleným osobám aby se dostali k tajnému obsahu, jako jsou osobní data nebo elektronická pošta.

Existují protokoly, které používají asymetrické kryptografie a protokoly, které používají symetrické kryptografie. Tento způsob znamená, že obě strany používají dva různé klíče.

- **SSL –** Tento protokol využívá šifrovací klíče a používá asymetrickou kryptografii. Klíče existují dva (veřejný a soukromý). Veřejný klíč je volně k dispozici a lze s ním zašifrovat data, které může pouze rozšifrovat majitel veřejného klíče, které rozšifruje svým soukromým klíčem. Komunikace mezi klientem a serverem využívající tento protokol se nejdříve obě strany dohodnout (server zašle klientovi certifikát a ten si klient ověří autentičnost). Poté se obě strany dohodnou jaký šifrovací klíč budou používat. Tímto klíčem potom bude šifrovaná jejich veškerá komunikace.

- **TLS –** Tento protokol je nástupcem výše zmíněného. Funguje na podobném způsobu.

- **GPG (GNU Privaci Guard) –** Tento protokol využívá asymetrického šifrování. Tento protokol se nevyužívá jen pro šifrování, ale i pro ověřování digitálního podpisu. GPG nepoužívá patentované algoritmy. Možnost také použití symetrické kryptografie.

2.6 Komprese datového přenosu

Pro zmenšení datového toku se využívá komprese, jejíž úkolem je zmenšit velikost datového obsahu souboru. Existují dva základní typy komprese – ztrátová a bezztrátová. K přenosů informací, které musí být všechny zachované se používá bezztrátová komprese.

Účinnost komprese bývá závislá na charakteru dat, které chceme zkomprimovat. Pokud tyto data v sobě obsahují opakující se vzory, pak je kompresní účinnost větší. Pokud jsou ale data náhodná a nelze v nich nají žádné opakující se vzory, pak je kompresní účinnost horší a v některých případech se nevyplácí komprimovat. Například v případě že po zkomprimování je výsledná velikost jen o 1,5% menší pak se komprese nevyplácí a jen to byla ztráta času.

Mezi bezztrátové komprese patří:

- **RLE –** Tento kompresní algoritmus využívá mnohonásobného výskytů sekvence, které zkracuje tak, že zapíše počet, kolikrát se sekvence vyskytuje. Například sekvenci AAAABBAA zapíše jako 4A2B2A. Účinnost tohoto algoritmu je závislá na charakteru dat a nejlépe se hodí na data, kde se často opakují hodnoty.

- **LZW –** Při kompresi se používá seznam pomocných frází.

- **Deflate –** Pokročilejší algoritmus komprese využívající slovník. Slovník napomáhá určité vzory popsat zkrácenou formou.

- **LZMA –** Algoritmus využívající slovníkové komprese, velikost slovníku je teoreticky neomezená. Jedná se o vylepšení algoritmu Deflate. Kompresní účinnost oproti Deflate je mnohem větší.

Mezi ztrátové komprese patří:

- **JPEG –** Kompresní formát obrázku využívající algoritmus kosinové tranformace.

- **H264 –** Formát kódování videa.

- **HEVC –** Kompresní videoformát využívající vysoce účinný algoritmus komprese podobný H264, který má dvojnásobně lepší kompresní poměr než H264, výstup je ve stejné kvalitě jako u H264 ale při polovičním datovém toku.

V kapitole Ztrátové kompresní formáty videa jsou detailněji popsané nejpoužívanější kompresní metody.

2.7 Způsoby detekce a oprav chyb během datového přenosu

K ověřování integrity dat, které byly doručeny po síti, nebo jiným druhem přenosu, se používají různé metody:

- **Kontrolní součet –** Jeho cílem je vytvořit ze souboru unikátní otisk (Hash), který je vytvořen na základě algoritmů. Kontrolní součet obvykle bývá distribuován vedle souboru, pokud si uživatel vytvoří sám kontrolní součet ze staženého souboru pomocí určitého programu, jehož musí shodovat s přiloženým kontrolním součtem. K vytvoření kontrolního součtu se používají hashovací funkce. Hashovací funkce vytváří speciální kód, který se nesmí za žádnou cenu při různých datech opakovat, jinak dojde ke kolizi. Kolize hashovací funkce znaméná, že když při různých stupech dochází v výstupu stejné hodnoty. Pomocí kontrolního součtu lze zjistit i drobné změny v souboru oproti originálu, drobné změny na vstupu můžou způsobit velké změny na výstupu v kontrolním součtu.

- **Parita –** Ověřování parity znamená že se odešle paritní bit, který označuje druh parity. V případě liché parity je paritní bit nastaven na 1 pokud odeslaná zpráva obsahuje lichý počet jedniček. Pokud je počet jedniček sudý pak je paritní bit roven nule. Existuje i ověřování sudé parity, ta počítá se sudým počtem bitů

- **Modulo –** Tento způsob využívá operace kdy se celočíselným dělením vypočte zbytek po dělení. Z celého datového rámce se sečtou všechny hodnoty všech Bytů a následně se tato hodnota podělí číslem 255 a z výsledku se použije zbytek po dělení.

2.7.1 Algoritmy kontrolního součtu

- **MD5 –** Vytváří zprávu otisku souboru o velikosti 128 bitů.

- **SHA –** Existuje několik verzí, např: SHA128, SHA256, SHA512. Tyto verze se liší podle počtu bitů, které používají pro tvorbu výstupu.

- **Blowfish –** Oproti výše zmíněným tento algoritmus používá navíc i klíč, pomocí něhož se na datech provede kontrolní součet, čímž se znemožní útočníkovi zpětně získat ze součtu původní data metodou hrubé síly, tedy systematickým testováním možných kombinací.

Tyto výše zmíněné algoritmy se používají na ověření stažených dat z internetu nebo pro ukládání hesel v databázi, kdy není bezpečné ukládat heslo v surové podobě a tak se ukládá ve formě kontrolního součtu. Pokud se tedy uživatel přihlásí, heslo které zadal se převede na kontrolní součet a ten se porovná se součtem uloženým v databázi.

2.7.2 Kontrola parity

Kontrola parity se používá k ověření počtu lichých nebo sudých bitů. K tomuto se používají dva typy parity: sudá a lichá. Tento druh kontroly se používá při komunikaci RS232, kde se zvolí jestli chceme kontrolu parity provádět nebo ne.

- **Lichá parita –** V případě že je nastavená kontrola této parity pak je paritní bit roven jedné tak aby celkový počet jedniček (nastavených bitů) byl ve zprávě lichý. Příkladem je první řádek

(příklad A) v Tabulce 2, kde data obsahují sudý počet jedniček a proto je paritní bit nastaven na hodnotu 1, počet jedniček v datech je 4, což je sudé číslo, proto je paritní bit roven jedné aby celkový počet jedniček byl lichý.

- **Sudá parita –** V případě že je nastavená kontrola této parity pak je paritní bit roven jedné tak aby celkový počet jedniček (nastavených bitů) byl ve zprávě sudý. Příkladem je druhý řádek (příklad B) v Tabulce 2, kde je v datech lichý počet jedniček, a z tohoto důvodu je paritní bit roven jedné při nastavení parity na sudou. Počet jedniček v datech je 7, což je liché číslo a proto je paritní bit roven jedné aby celkový počet jedniček byl sudý.

Příklad použití paritního bitu:

Tabulka 2: Příklad použití paritního bitu

Příklad	DATA	Paritní bit	
		Sudá parita	Lichá parita
A	01001101	**0**01001101	**1**01001101
B	10111111	**1**10111111	**0**10111111

2.7.3 Algoritmus Modulo

Jedná se o typ operace, při níž je vyhodnocen zbytek po celočíselném dělení. Číslo modulo může být uvedeno na konci řetězce zprávy, které udává jednoduchý kontrolní součet jehož velikost se vleze do jednoho Bytu. Přijímací strana si může modulo z přijatých dat snadno odvodit a porovnat s modulem, které bylo přijato na konci zprávy. Poku obě modula souhlasí pak je zpráva kompletní. Operaci modulo značíme znaménkem %.

Příklad:
Máme data, která chceme odeslat:
Data: 12, 200, 100, 128
Pokud chceme vytvořit kontrolní součet pomocí modula nejdříve musíme sečíst všechny hodnoty Bytů.
Součet = 12 + 200 + 100 + 128 = 440
Vzniklý součet je příliš velký na to aby se vlezl do 1 Byte, proto provedeme operaci modulo %255.
Modulo (440 % 256) = 184.
Tím vznikla hodnota o kolik přetekla hodnota v Bytu.

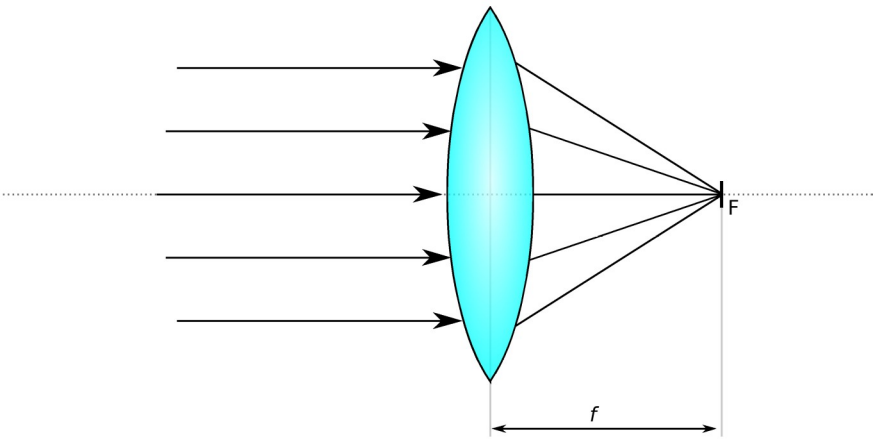
3. Rozbor problematiky zpracování obrazového signálu.

Zpracování obrazového signálu je tvořeno několika základními operacemi. Obraz je z počátku tvořen světelnými paprsky, které jsou čočkou ohýbány a zaostřován, tak aby vznikl ostrý obraz na ploše, kde dopadá.

Obraz před jeho zpracováním je nejdříve zachytáván pomocí kamery. Kamera je zařízení schopné zaznamenat plynulý video obraz v určitém rozlišení a v určité kvalitě.

Kamera se skládá ze soustavy čočky a clony, které usměrňují tok světla, tak aby dopadal ostrý na snímací plochu, která je tvořená CCD čipem nebo filmem.

- **Čočka** je většinou skleněný útvar schopen ohnout paprsek světla. U čočky se udává ohnisková vzdálenost, což je číslo, které udává vzdálenost bodu od čočky do které je světlo z teoretického nekonečna promítnuto.



Obrázek 10: Znázornění ohniskové vzdálenosti

- **Clona** je kruhový otvor s nastavitelnou velikostí, slouží k redukci toku světla a ke změně hloubky ostrosti.

- **CCD snímač** je polovodičová součástka stávající se z velkého počtu drobných pixelů, které po dopadnutí světla získají náboj, který se následně přečte z paměťového registru CCD snímače. Při přivedení napětí na CCD snímač se stává citlivý na světlo a je schopen nechat fotony excitovat elektrony v polovodiči.

- **Film –** Ve starších dobách se používal film z plastu nebo celulózy, na němž je nanesená světlocitlivá vrstva obsahující například dusičnan stříbrný. Film po osvícení změní své chemické vlastnosti a musí se dát do vývojky, aby se vyvolal zaznamenaný obraz a následně se použije ustalovač, který zabrání dalším reakcím na světlo. Takto vyvolaný film vytvoří negativ (barvy jsou obrácené oproti skutečnosti)

Mezi klíčové parametry při zachytávání videa patří rychlost snímků za sekundu a rozlišení.

- **Snímková frekvence (FPS) –** Udává rychlost snímků za sekundu, kterou je pořizováno video.
- **Rozlišení –** V případě pořizování digitálního záznamu se uvádí i rozlišení videozáznamu (horizontální a vertikální velikost matice pixelů).

3.1.1 Způsoby ukládání obrazu

Obraz může být uložen jak v digitální podobě, tak i analogové podobě. Co se týče analogové podoby nabízejí se tyto možnosti:

- **Film –** Analogový způsob uchování obrazu, který vzniká na světlocitlivé vrstvě působením dopadajícího světla. Výsledkem je negativ.
- **Magnetická páska –** Tento způsob využívá magnetické stopy zapsané na pásku na jejíž povrch je nanesená feromagnetická vrstva. Obraz je zde zaznamenán podle normy PAL nebo NTSC. Tyto dvě normy definují přenosové principy, které využívají frekvenční pásma ve kterých se zapisují různé hodnoty barev.

Pokud je obraz uložen v digitální podobě, bývá obvykle zakódován v nějakém formátu a je uložen na nějakém nosiči (CD, HDD, Flash paměť)

3.1.2 Popis digitální podoby obrazu

Video obraz je tvořen jednotlivými snímky, které se mění v čase rychlostí jenž se nazývá snímková rychlost. Každý snímek je tvořen body jenž se nazývají pixely. Každý pixel má daný počet variací barev a tento počet se určuje podle dané barevné hloubky.

Pixely tvoří uspořádanou matici jejíž rozlišení je pro každý snímek ve videu stejný. Rozlišení matice je dáno násobkem sloupců a řádků.

U obrazu rozlišujeme tyto parametry

- **Barevný prostor obrazu –** Udává informace o významu jednotlivých barevných složkách v pixelu.
- **Barevná hloubka –** Udává maximální počet barevných variací v snímku.
- **Rozlišení obrazu –** Rozlišením obrazu je myšleno rozlišením matice ze které se obraz sestává.
- **Snímková rychlost –** Počet snímků za sekundu.

3.1.2.1 Barevné prostory obrazu

Každý snímek je uložen v určitém barevném obrazu. Každý pixel se skládá z subpixelů, což jsou jednotlivé části nesoucí hodnotu jedné barevné složky, obvykle bývají tři. Barevný obraz udává způsob uložení barvy do jednoho pixelu, jak se chovají jednotlivé subpixely a které složky barev nesou.

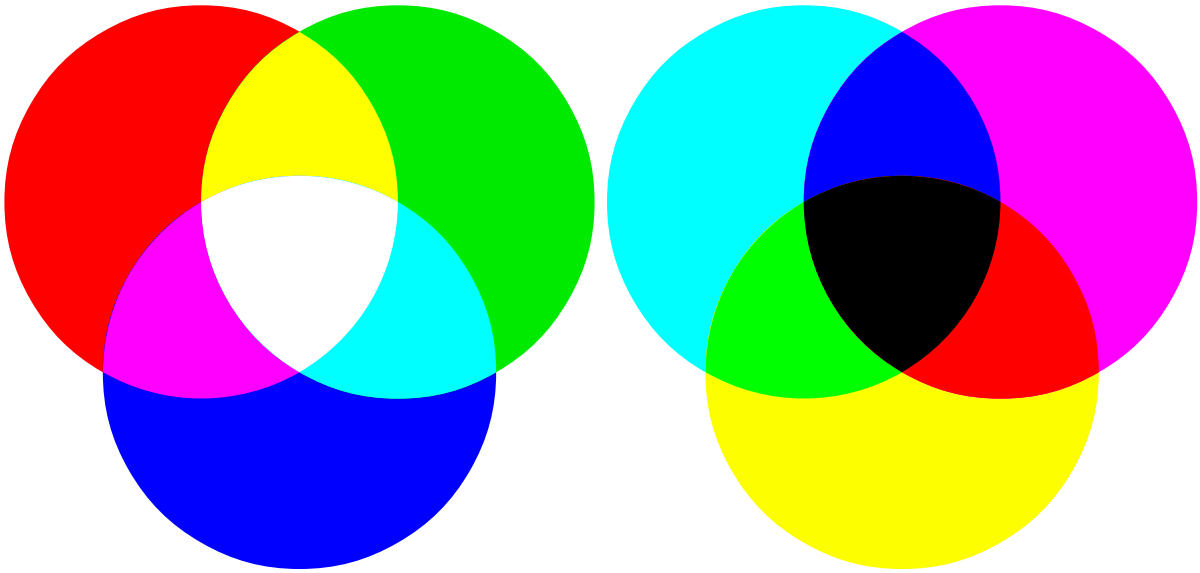
V případě barevného obrazu pixely mohou být tvořeny subpixely v nichž jsou uloženy jednotlivé složky barvy. Rozlišení počtu barev udává bitová hloubka pixelu. Běžně používaná barevná hloubka je 24 bitů, což znamená že každá barevná složka má celkem 8 bitů (celkem 256 barevných možností na jeden subpixel).

- **RGB –** Tento prostor vychází z přirozeného skládání barev na běžném LCD monitoru. Obraz je složen třemi složkami: červená, zelená, modrá. Jsou i jiné varianty (BRG, BGR, atd), které jen popisují pořadí barev v subpixelech. Existuje varianta RGBA, kde je navíc přidán alfa kanál (průhlednost).

- **YUV –** Obraz tvoří tři složky: Y – jasová složka, U - (modrá až žlutá) a V - (červená až žlutá) jsou barevné složky.

- **YcrCb –** Obraz tvořen třemi složkami, Y – jas, Cr – modrá složka, Cb – červená složka.

- **CMYK –** Barevný prostor používaný při tisku. Obsahuje celkem 4 složky (azurová, purpurová, žlutá, černá). Míchání barev oproti RGB je opačné, nesčítá se hodnoty světel, ale barvy světlo naopak filtrují.



Obrázek 11: Porovnání RGB (vlevo) a CMYK (vpravo) barevného modelu se znázorněním skládání barev.

3.1.2.2 Barevná hloubka snímků

Každá barevná složka je zakódován určitým počtem bitů. Tím se určuje jeho přesnost podání barev. Čím vyšší je počet bitů na složku, tím je variabilita barev větší.

Tabulka 3: Barevné hloubky pixelů

Typ	Bity na pixel	Bity R	Bity G	Bity B	Počet barev	Rozsah na jednu složku
int	8	3	3	2	256	0-7
int	24	8	8	8	16777216	0-255
int	48	16	16	16	2^{48}	0-65535
int	96	32	32	32	2^{96}	0-4 mld
float	96	32	32	32	32	HDR

Dle barevné hloubky lze určit maximální počet variací barev z následujícího vztahu:

$$2^{BPP}$$

Kde *BPP* je počet bitů jimž je tvořen jeden pixel. Počet bitů na pixel získáme vynásobení počtu barevných složek a počtem bitů jimiž je složka tvořená.

$$BPP = počet\ složek \cdot počet\ bitů\ na\ složku$$

V případě barvy jenž je tvořená třemi složkami po osmi bitech (RGB888) pak bude výpočet následující.

$$2^{3 \cdot 8} = 2^{24} = 16777216$$

U tohoto barevného typu je maximální počet barevných variací větší než 16 milionů.

3.1.2.3 Rozlišení snímku

Digitální obraz je tvořen maticí pixelů, přičemž každý pixel se obvykle skládá ze tří barev. Matice obrazu má definované rozlišení, které se udává v počtech pixelů na řádek krát počet pixelů ve sloupci, například 640x480.

Rozlišení snímku je definované na základě počtu pixelů v řádku a ve sloupci. Velikost snímku v bytech se liší podle barevného typu snímku. Pokud máme snímek, který má 8 bitů na barevnou složku platí následující. U černobílých snímků je pixel velký pouze jeden Byte, neboť v sobě nenese žádné informace o barvách, ale pouze má jednu informaci o jasu.

U barevných snímků je velikost jednoho pixelů vetšinou 3 Byty. Je to dáno tím že obsahuje tři dílčí složky velikostně po jednom Byte.

Tabulka 4: Standardní rozlišení

Standart rozlišení	Šířka v Px	Výška v Px
Full HD	1920	1080
HD	1280	720
SXGA	1280	1024
SVGA	800	600
VGA	640	480

Výpočet velikosti obrázku:
Velikost obrázku se vypočte podle jeho rozměru matice která se vypočte podle vztahu:

$$(počet\ pixelů\ na\ řádek \cdot počet\ bytů\ na\ pixel) \cdot počet\ řádků$$

Pokud máme RGB obrázek s rozlišením 640x480 (neuvažujeme o jeho komprimované formě), tak jeho teoretická velikost bude následující.

$$(640 \cdot 3) \cdot 480 = 921600\,B$$

Velikost je udávána v Bytech. Pokud by takové video bylo přenášeno například se snímkovou frekvenci 25 snímků za sekundu, jedna minuta tohoto videozáznamu by zabrala velikost 1318 MiB. Tato velikost se spočte vynásobením velikosti jednoho snímku krát snímková frekvence krát počet sekund.

$$velikost\ snímku\ v\ bytech \cdot FPS \cdot počet\ sekund$$

$$921600 \cdot 25 \cdot 60 = 1382400000\,B$$

$$\frac{1382400000}{1024} = 1350000\,kiB$$

$$\frac{1350000}{1024} = 1318\,MiB$$

Tato velikost je pro přenos v síti nepřípustná, proto se používají kompresní metody pro zmenšení objemu dat při vysílání videa.

3.1.2.4 Snímková frekvence

Snímková frekvence udává počet přehraných snímků za sekundu. Obvykle se udává v jednotkách FPS (anglicky frames per second) nebo v hertzích. Snímková rychlost se udává u pohyblivého obrazu například v televizním vysílání. Princip pohyblivého obrazu stojí na základě, že lidské oko je schopno při vyšších frekvencích vnímat jednotlivé snímky jako plynulý pohyb. Obraz s frekvencí menší než 15 Hz je zdravé lidské oko schopno vnímat jako jednotlivé individuální snímky a vytvoří dojem „trhaného obrazu“.

Běžné snímkové frekvence, které se používají v 21. století jsou například: 25 Hz, 24Hz, 50 nebo 60Hz.

3.2 Rozbor kompresních metod obrazu

Ke zmenšení objemu datového toku videa se používají kompresní algoritmy. Nejčastěji se používá ztrátová komprese. Tento typ komprese se snaží odstranit drobné detaily, které lidské oko nevnímá a není třeba je ponechat v tomto obrazovém vysílání.

Ztrátová komprese produkuje výstup s mnohem menší velikosti oproti bezztrátové. Tento druh komprese sice redukuje některé detaily, které nejsou důležité pro lidské oko.

3.2.1 Obecný princip ztrátové komprese videa

Prvním procesem při ztrátové kompresi je transformace dat na frekvenční oblast za pomocí diskrétní Fourierové transformace nebo diskrétní Kosinové transformace nebo diskrétní vlnkové transformace.

Následně se ve vniklém spektru rozhodne o ztlumení, nebo odstranění některých frekvenci, které lidské oko není schopné vnímat. Kompresní algoritmy rozhodují o tom jaké frekvence jsou důležité a které ne. Tento proces neprobíhá na celém snímku, ale snímek se rozdělí nejprve na makrobloky, což jsou jednotlivé části snímku, na kterých se poté provede výše uvedený proces.

V některých kompresních algoritmech se používá jiný barevný prostor než RGB, například YUV. Důvodem je, že u prostoru YUV (který se skládá z jedné složky jasové a ze dnou barevných) můžou být jeho barevné složky zmenšeny a v plném rozlišení se ponechá pouze jasová složka.

Snímky jsou kódovány po skupinách (GOP – group of pictures), což je skupina snímků dané velikosti, která je ohraničena dvěma kompletními snímky (intra frames) a mezi němi se nachází mezisnímky (inter frames), které popisují pouze rozdíly v pohybu zaznamenaných objektů.

O dalších parametrech a rozhodování o psychovizuálních modelech rozhodují jednotlivé algoritmy, které jsou implementované v kompresních formátech (například H264, DivX, VP8).

3.2.2 Ztrátové kompresní formáty videa

Kompresní formáty se rozlišují podle algoritmů, které rozhodují o parametrech, jejichž hodnoty určují výsledný způsob zakódování videa. K převodu videa na tyto formáty se používá kodér. Mezi nejvíce používané formáty ztrátové komprese videa patří následující.

- **H264 –** Tento formát používá algoritmus, který snímek rozdělí na menší části, jejichž velikost se určí podle potřeby. Snímky jsou rozděleny do skupin (GOP) (ve výchozím nastavení 12 snímků). Snímky jsou navíc rozděleny na makrobloky o velikosti 16x16 pixelů. Celý snímek je převeden na barevný prostor YUV nebo podobný (YCbCr), který používá první část jako jasovou složku. Kvalita kódování je ovlivněna uživatelsky definovaným datovým tokem, který určuje dostupný prostor pro kodér. Čím vyšší je datový tok, tím lepší je kvalita obrazu. Nízký datový tok totiž omezuje zpracování vyšších frekvencí, což má následek, že se výrazně ztratí

kvalita malých vzorů (např listí, vlny, tráva) a vede k jejich rozmazání. Podporuje však variabilní bitrate (VBR), což znamená že se datový tok umí automaticky přizpůsobit tak, aby byla na výstupu zachována konstantní kvalita. Kodér používající algoritmus tohoto formátu má na výběr použití uživatelem stanoveného datového toku. Nebo namísto volby datového toku lze použít parametr CRF (constant rate factor) jehož hodnota udržuje konstantní kvalitu a kodér se snaží automaticky přizpůsobit datový tok aby byla zachována konstantní kvalita.

- **Theora –** Tento formát je licenčně nezatížen, je svobodný. Podporuje taktéž VBR a jeho komprese je založená na diskrétní kosinové transformaci.

- **DivX / Xvid –** DiviX je formát založený na podobném standardu jako H264, je patentově zatížený a z tohoto důvodu vznikl formát Xvid, který je svobodný.

- **VP8 –** Formát využívající algoritmy s podobnou účinnosti jako H264.

- **VP9 –** Formát jenž má algoritmy 2x účinnější než VP8, účinnost má přibližně stejnou jako H265

- **H265 –** Formát založený na podobném základě jako H264, ale s 2x účinnější kompresní účinnosti. Ve srovnání s H264 má obraz stejnou kvalitu i při polovičním datovém toku. Ale algoritmus pro kódování je složitější a tím je náročnější pro výkon.

3.2.3 Kontejnery používané pro uložení multimediálních proudů

Samotný kodér je schopen vyprodukovat pouze data které obsahují pouze jednu stopu videa nebo audia, což není vhodné pro přehrávání záznamů, protože zvuk i video by byly uloženy zvlášť. Je třeba tyto stopy nějakým způsobem spojit. K tomuto účelu slouží multimediální kontejnery.

Kontejnery v sobě nesou informace o jednotlivých stopách jako je typ stopy, délka stopy a formát v němž je stopa zakódována. Kontejnery taktéž podporují i zabudované stopy titulků. Některé kontejnery podporují názvy stop (jazyk zvukové stopy).

Mezi nejčastější kontejnery patři:

- **AVI –** Tento formát se dělí na tři části, v první části je hlavička, která obsahuje informace o audio nebo video stopách jako jsou kompresní algoritmy a snímková frekvence videa a počet kanálu zvuku. Druhá část obsahuje samotná data, sekvence videa a zvuku. Třetí část obsahuje AVI index, což je informace o časové poloze jednotlivých stop. Vzhledem k tomu že tento kontejner obsahuje hlavičku s podstatnými informacemi už na začátku, lze takovýto soubor přehrávat i když není kompletně stažen.

- **MP4 –** Tento formát také podporuje mnoho formátů avšak o něco méně než MKV a má jistá omezení. Navíc nepodporuje streaming. Taktéž podporuje titulky.

- **Matroska (MKV) –** Tento kontejner podporuje nejvíc možných formátů a podporuje teoreticky neomezený počet stop. V praxi se dá použit pro tvorbu filmu, který má více jazykových zvukových stop.

- **OGG –** Tento kontejnér vznikl hlavně pro formáty theora a vorbis. Ale podporuje i některé formáty z rodiny MPEG jako je H264.

- **Webm –** Kontejner založený na kontejneru Matroska (MKV). Určený pro přehrávání v prohlížeči v HTML5 elementech. Podporuje VP8 a zvuk Vorbis.

4. Návrh síťového kamerového systému

Síťový kamerový systém s datovým přenosem obrazového signálu je navržen, tak aby měl být schopen v síti vysílat a přehrávat libovolný počet kamer. Počet příjemců obrazového vysílání je teoreticky neomezený, z důvodu používání skupinových adres. Uživatel může nastavovat, které kamery chce v síti vysílat, uživatel si zvolí připojené kamery dá jím jméno a potvrdí, že je chce zveřejnit. Zároveň má uživatel možnost si vybrat, které dostupné přenosy v síti on sám chce sledovat a zobrazit je jako miniatury na které může kliknout a zvětšit si je. Uživatel také může u některých miniatur zapnout detekci pohybu. Při zjištěném pohybu na některém obrazovém přenosu se jeho miniatura zvětši na celou obrazovku. Lze také vybrat prioritu detekce pohybu, pokud bude pohyb zaznamenán na vícero živém vysílání videí pak se zvětší miniatura u níž je větší priorita.

Při přenosu je důležité dbát na to , aby celková rychlost datového přenosu nepřekročila maximální možnou přenosovou rychlost sítě. Tím se docílí kompresí obrazu a zvuku. A síť by neměla být zatěžována opakováním paketů, které nedorazili k cíli, proto se pro tyto účely hodí použít síťový protokol UDP. Pokud se použijí skupinové adresy (tzv. multicast) dojde ke snížení datového toku při vysílání zvuku a obrazu, neboť se nemusí posílat data každému příjemci zvlášť. Pokud se tato řešení aplikují, měl by se celkový datový tok v síti zmenšit.

4.1 Principy použité v návrhu kamerového systému

Architektura Klient-Server: Tento princip je použit pro výměnu informací mezi kamerovými zařízeními, které z pohledu z této architektury fungují jako klienti. Serverem je zde služba obsahující seznam kamer a jejich skupinové adres. Každý kamerový systém si od hlavního serveru vyžádá seznam dostupných kamer s jejich IP adresami na kterých se vysílají multimediální data. Server funguje pouze jako databáze IP adres.

Skupinové vysílání (multicast). Každé kamerové zařízení vysílá video přenos po síti na skupinových adresách (224.0.0.0 – 239.255.255.255). Tyto skupinové adresy se nepřidělují DHCP serverem, ale každá jednotka si pro svou kameru musí vybrat tuto adresu sama. Pomocí serveru si pak jednotky najdou seznam obsazených adres a podle něj si vyberou volnou adresu pro své kamery, které ještě neměly přidělenou svou adresu.

Zachytávání a zpracování obrazu. Obraz je zachytáván pomocí USB kamer podporující pořizování plynulý záznam. Tento záznam je softwarově nahráván pomocí knihovny OpenCV, což je knihovna metod, které se zabývají zachytáváním a zpracováním obrazu a detekcí tvarů. Následně je obraz zpracován enkodérem, který používá ztrátový kompresní algoritmus, což má za následek zmenšení datového toku na požadovanou velikost. Velikost datového toku je nejdůležitější požadavek na přenos v síti. Cílem je zvolit kompromis mezi kvalitou a datovým tokem, kde datový tok nesmí být příliš vysoký nebo příliš nízký.

Využití IP protokolu. Jednotlivá zařízení mezi sebou komunikují pomocí IP protokolu, kde jsou přiřazeny IP adresy, díky nímž se mohou navzájem kontaktovat (např. jednotka pošle serveru svůj seznam adres s kamerami). Díky IP adresám, lze v síti rozlišit jednotlivá zařízení což usnadní například komunikaci se serverem, který má pevně nastavenou statickou IP adresu, což usnadní jeho vyhledávání.

4.2 Metoda využití severu jako centrálního uzlu

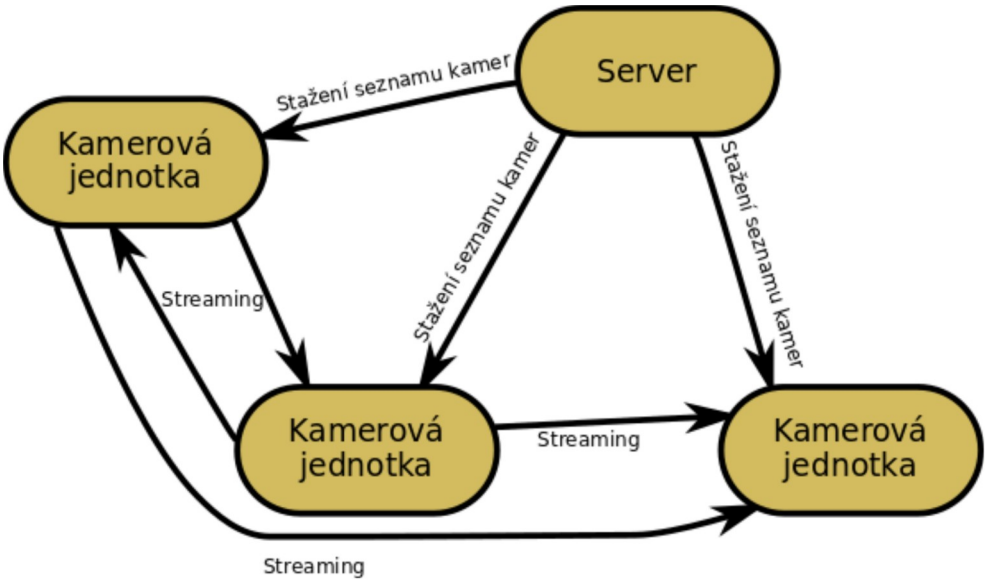
Tato metoda uvažuje využití serveru jako centrálního uzlu určenému k přeposílání všech živých vysílání v síti. Každý vysílací síťový uzel by posílal obrazová data na server a každý přijímací uzel by tato data přijímal. S rostoucím počtem vysílacích a přijímacích uzlů připojených ve stejný čas na server, by se zvyšovalo zatížení serveru.

4.3 Metoda síťového přenosu s IP kamerami

Tato metoda používá zapojení IP kamer do sítě. Ty však umožňují pouze vysílat obraz a nikoliv jej i přijímat a zobrazovat. V síti by pak museli být kromě IP kamer i zobrazovací zařízení.

4.4 Metoda IP multicastingu

Tato metoda taktéž využívá server v místní síti, ale pouze k předávaní vysílacích skupinových adres. K samotnému vysílání se používají skupinové adresy (IP multicast).



Obrázek 12: Komunikace mezi kamerovými jednotkami a serverem

4.5 Návrh software v kamerovém sytému na straně multimediálních jednotek

Software je navržen, tak aby uživatel mohl sledovat více multimediálních vysílaných obrazů najednou. Jednotky mohou mezi sebou přenášet video obraz po síti prostřednictvím skupinových adres (IP multicast). Mohou zároveň vysílat i zobrazovat vysílání v síti. Tato metoda byla vybrána pro tuto práci.

Každá multimediální jednotka se připojuje do sítě ve které je přítomen databázový server se seznamem adres na kterých se živě vysílá video. Tyto jednotky si mezi sebou přeposílají data obsahující živý obraz i zvuk na skupinových (multicastových) adresách viz obrázek 12 a 13.

4.5.1 Zachytávání pohyblivého obrazu

Zachytávání pohyblivého obrazu neboli videa je řešeno prostřednictvím web kamery zapojené do vestavěného systému na němž běží aplikace pro zachytávání a zpracování obrazu. Obraz je zachycen v podobě sekvence barevných matic, které mají pevně danou snímkovou rychlost.

4.5.2 Zpracování zachyceného videa

Video je po jeho zachycení nejdříve zmenšeno na požadovanou velikost, kterou zadá uživatel v grafickém rozhraní. Poté je video obraz převeden do barevného prostoru, který umožní snadnější kódování pro kodér využívající tento barevný prostor. Předposledním krokem je převedení videa na daný komprimovaný formát H264, který byl vybrán jako kompromis mezi ostatními formáty neboť zvládá přijatelnou kvalitu při datovém toku který není příliš vysoký na to aby zatěžoval síť. Posledním krokem je samotné vysílání po síti. Vysílaná data jsou směrovány na skupinovou adresu, odkud mohou všichni příjemci sledovat video přenos.

4.5.3 Princip jištění dostupných kamer v síti

Aby se jednotlivé multimediální jednotky mohly mezi sebou najít a komunikovat spolu a sledovat živé vysílání videa musí navzájem znát své adresy.

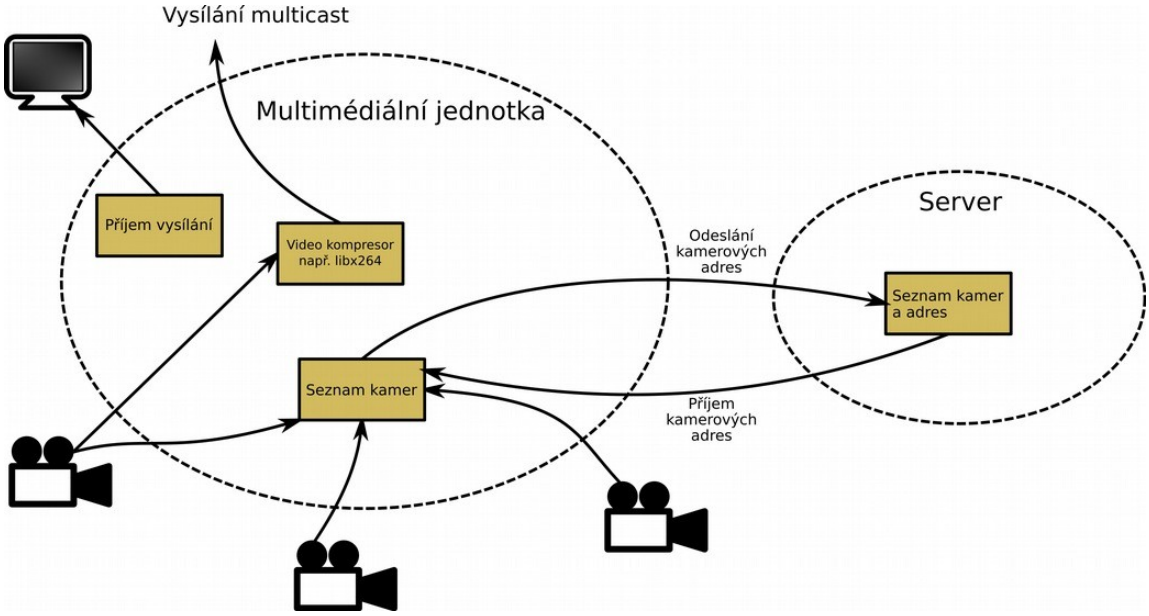
Po připojení této jednotky do sítě jíž je známa adresa serveru se tato jednotka připojí na databázový server a zašle jemu svůj seznam kamer se skupinovými adresami přiřazeny kamerám. Poté tato jednotka požádá server o zaslání kompletního seznamu adres, které byly serveru zaslány jinými jednotkami v síti které se připojili dříve než tato jednotka. Tato komunikace používá TCP protokol.

Nyní již jednotka má seznam známých kamer v sítí a uživatel si může vybrat, které chce sledovat.

4.5.4 Princip přenosu živého vysílání mezi multimediálními jednotkami

Přenos živého vysílání je zprostředkován přenosem UDP datagramů na skupinových adresách. Tyto adresy si každá jednotka zjistí ze serveru, aby na nich mohla sledovat živé vysílání obrazu a zvuku.

Pokud jednotka má za úkol vysílat obraz ze své připojené kamery, nejdříve musí zjistit které skupinové adresy jsou již obsazené. Skupinová adresa na které se bude vysílat se zvolí tak že se zvolí adresa z rozsahu 224.0.0.0 – 239.255.255.255, která není uvedena v seznamu použitých adres získaného ze serveru.



Obrázek 13: Schéma systému kamerové jednotky a serveru

4.5.5 Detekce pohybu na přijatých videích

U sledovaných video přenosů si uživatel může vybrat zda chce povolit detekci pohybu. Detekce pohybu spočívá v porovnávání jednotlivých snímků. Porovnávání snímků probíhá způsobem při kterým se porovnávají jednotlivé pixely. Tento způsob může být výkonově náročný, ale nároky lze snížit převedením snímků na černobílé a následným rozdílem je porovnat. Použitím prohového filtru lze nastavit citlivost detekce a zabránit tak aby byl šum ve videu interpretován jako pohyb.

4.6 Návrh software kamerového systému na straně serveru.

Software na straně serveru je navržen tak aby přijímal požadavky klientů (multimediálních jednotek) a odpovídal na ně. Server umí plnit celkem dva požadavky: první je registrace nové adresy, kde klient

zašle registrační příkaz s adresou. Druhý příkaz je požadavek na server aby klientovi, který tento požadavek zaslal, poslal seznam všech adres, které na server uložili ostatní klienti. Server má uložené pouze skupinové adresy kamer a jejich názvy zvolené uživatelem.

Server pří jímá dva typy žádostí:

Žádost o získání seznamu adres:

1. Klient vyšle na server žádost o seznam adres.

2. Server klientovi odpoví seznamem všech adres které má uložené.

Žádost klienta o uložení seznamu adres na server:

1. Klient vyšle žádost na server o uložení adresy ve formě zprávy jejíž součástí je název, skupinová adresa.

2. Server tuto zprávu rozdělí na části podle významu a uloží si předané parametry, včetně adresy klienta, která ve správě není, ale nachází se v protokolu TCP komunikace.

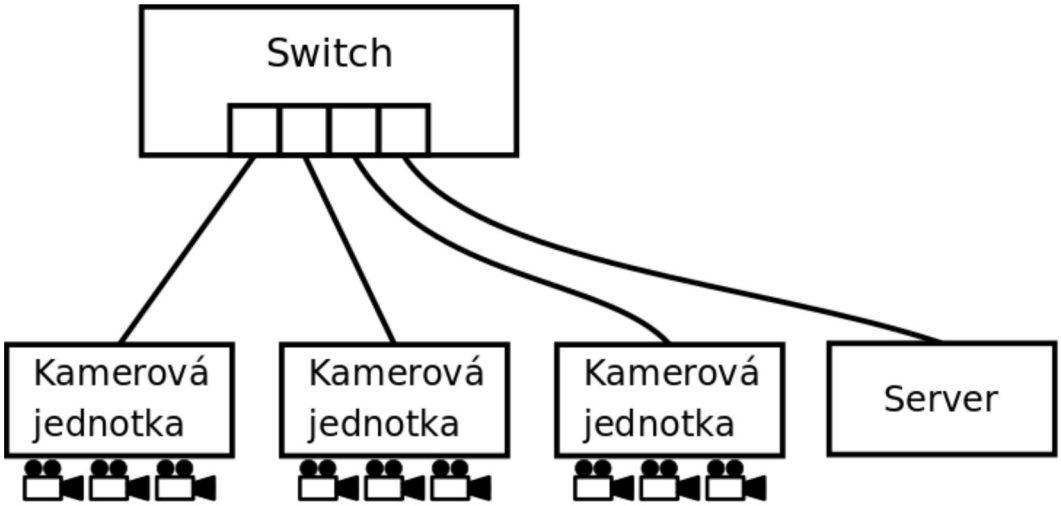
4.7 Návrh hardware pro multimediální jednotky a server

Hardware multimediálních jednotek je zvolen tak aby splňoval minimální požadavky pro práci v síti a zvládal kódování videa v reálném čase. Tento hardware má dostupné porty na zapojení kamer a síťový port pro připojení do sítě.

Použitý hardware je vestavěný systém, jehož jednotlivé součásti jsou grafický čip, operační paměť, procesor a síťový adaptér. Díky přítomnosti grafické karty je využito hardwarové kódování, které je většinou rychlejší než kódování s použitím CPU.

Hardware serveru je zvolen pouze pro síťové účely. Server nezpracovává obraz ani zvuk, proto nejsou kladeny na server vysoké nároky na výkon. Server pouze obsluhuje klienty, kteří pouze požadují seznam adres což jsou textová data s malým objemem. Komunikace mezi klienty a servery probíhá jen v určitých intervalech při kterých je přenášen malý objem dat, které v sobě zahrnují pouze názvy kamer a jejich adresy. K přenosu obrazových dat mezi serverem a klienty nedochází.

Každá multimediální jednotka je zapojená do sítě společně se severem a vytváří tak v síti kamerový systém s přenosem obrazového signálu i se zvukem.



Obrázek 14: Zapojení jednotek a serveru v síti

5. Realizace Síťového kamerového systému

Celý kamerový systém je složen z jednotlivých multimediálních jednotek v síti a jednoho serveru fungující jako databáze. Jednotlivé jednotky jsou vestavěná zařízení založená na architektuře ARM. Pro tyto multimediální jednotky bylo vybráno Raspberry Pi 3. Pro server bylo vybráno zařízení zvané Raspberry Pi Zero W.

K realizaci hardware pro multimediální jednotky byl vybrán Raspberry Pi 3, neboť má hardware dostatečně výkonný pro práci s multimediálními daty jako je audio a video. Jeho grafická karta umožňuje hardwarové kódování videa.

Raspberry Pi 3 je mikropočítač založený na architektuře ARM. Co se týče síťových adaptérů tak má ethernetový port a WIFI adaptér. Disponuje také technologii Bluetooth.

Disponuje čtyřmi USB 2.0 porty, které jsou použity v tomto projektu pro USB kamery. Kamer lze zapojit víc než čtyři v případě použití USB hubu.

Grafickou kartu má dostatečně výkonnou na to aby zvládala hardwarově kódovat video. Procesor má dostatečný výkon pro operační systém i s grafickým rozhraním. K dispozici je také HDMI port jenž umožňuje připojit LCD displej.

K dispozici je také slot pro MicroSDHC kartu, na níž může být uložen operační systém.

Raspberry Pi 3 je kompatibilní s operačním systémem Linux, díky této kompatibilitě je tento mikropočítač schopen spustit systémy jako jsou například Debian nebo Raspbian, který je na Debianu založený.

Tabulka 5: Hardwarové vlastnosti Raspberry Pi 3

Architektura	ARMv8
Paměť	1 GB
USB 2.0	4x
Video výstup	HDMI
Grafika	Broadcom VideoCore IV
Zvukový výstup	3,5mm jack, HDMI
Úložiště	Slot pro MicroSDHC
Síť	10/100 Mbit/s Ethernet, WiFi 802.11n, Bluetooth 4.1
Ostatní periferie	GPIO

5.1 Realizace hardware na straně serveru

Zařízení, které hraje roli serveru v síti je založeno na Raspberry Pi Zero. Raspberry Pi Zero je vestavěné zařízení, které je podobné Raspberry Pi 3, ale má mnohem menší výkon.

Raspberry Pi Zero má dvě varianty:

- **Raspberry Pi Zero:** nemá možnosti síťového připojení.
- **Raspberry Pi Zero W:** Disponuje Bluetooth a Wifi modulem.

V této práci byla použitá druhá varianta s Wifi modulem.

Ačkoliv tato varianta Raspberry nemá nijak velký výkon ve srovnání s Raspberry Pi 3, výkon je pro jeho účel dostačující. Server je využit pouze pro ukládání záznamů adres jednotlivých multimediálních přenosů. Do sítě se připojuje pomocí WIFI.

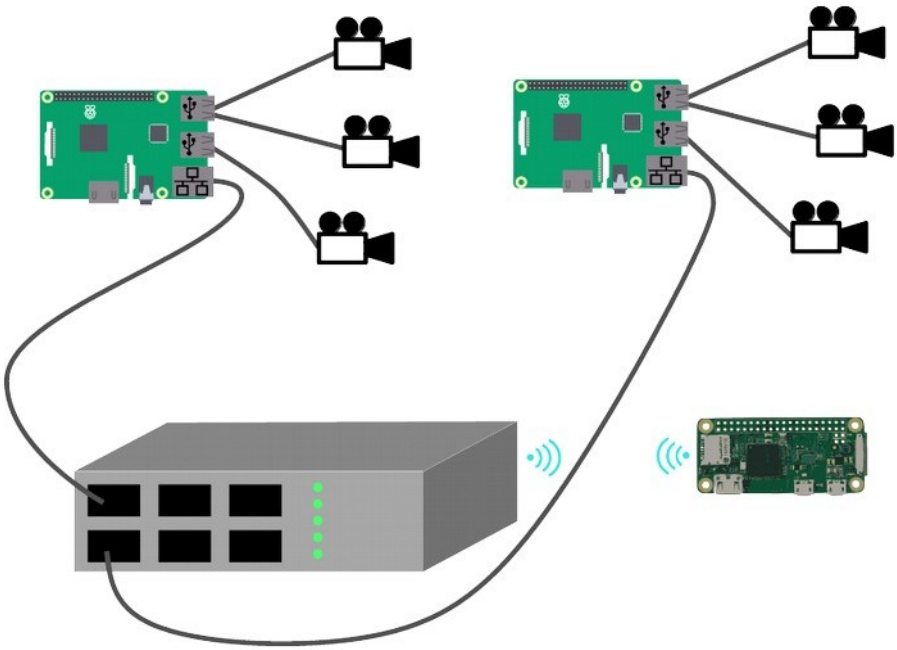
Tabulka 6: Hardwarové specifikace Raspberry Pi Zero W

Architektura	ARMv6
Paměť	512 MB
USB 2.0	MicroUSB 1x
Video výstup	mini-HDMI
Grafika	Broadcom VideoCore IV
Zvukový výstup	3,5mm jack, HDMI
Úložiště	Slot pro MicroSDHC
Síť	WiFi 802.11n, Bluetooth 4.1
Ostatní periferie	GPIO

5.2 Realizace zapojení multimediálních jednotek do sítě

Jednotlivé multimediální jednotky s kamerami jsou do sítě zapojeny přes Ethernet pomocí kroucené dvojlinky, zajistí se tím dostatečně rychlý a stabilní přenos. Centrálním prvkem je zde Switch, který rozšiřuje síť a díky němu lze připojit i více jednotek.

Příklad zapojení dvou multimediálních jednotek může být viděn na obrázku č. 15, který znázorňuje zapojení těchto jednotek do Switche. Každá z nich má v USB portech zapojené tři web kamery. Tyto multimediální jednotky jsou zapojeny kabelem a server založený na Raspberry Pi Zero W je připojen přes WIFI (vpravo dole).



Obrázek 15: Příklad použitého zapojení dvou multimediálních jednotek a jednoho serveru (Raspberry Pi Zero W) do routeru (Ilustrace)

5.3 Realizace software na straně serveru.

Softwarové řešení je postaveno na systému Linux. Program je napsaný v C++ což umožnilo mít program ve formátu nativního linuxového spustitelného souboru, který pracuje podstatně rychleji než interpretovaný kód (například v Pythonu nebo Perlu).

Server přijímá dva příkazy a je schopen detekovat odpojení klienta ze sítě a vymazat ze seznamu vysílací adresy jeho kamer. Příkazy jsou ve formě textového protokolu. Tyto příkazy vždy vysílá klient (multimediální jednotka).

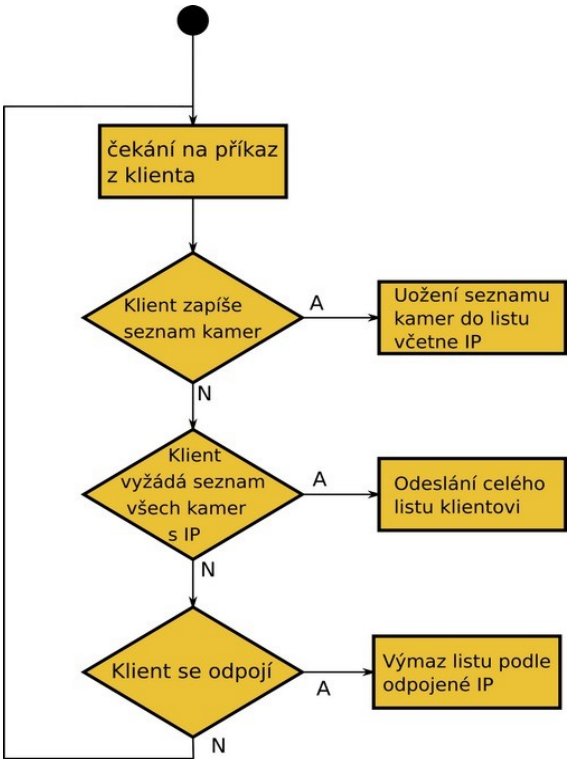
Tyto dva příkazy se jmenují *REG* a *GET*. Pokud server obdrží od klienta příkaz *GET*, server tomuto klientovi pošle seznam všech adres a názvů obsažených v listu založeném na struktuře *CameraInfo*. Pokud server obdrží příkaz *REG* a za ním hned následující skupinovou adresu s názvem kamery, server tyto parametry uloží do listu.

V následující tabulce jsou uvedeny příkazy které posílá klient.

Tabulka 7: Příkazy příjmané serverem

Příkaz	Význam	Příklad
GET	Získání seznamu	GET
REG multicast_ip jmeno	Registrace nové adresy	REG 224.0.0.1 doma

Při spuštění serverové aplikace server nejdříve čeká na nového klienta. Po připojení klienta server vyčkává na jeho příkazy, ty jsou rozlišené pomocí *if* a *else*. Viz obr. 16.

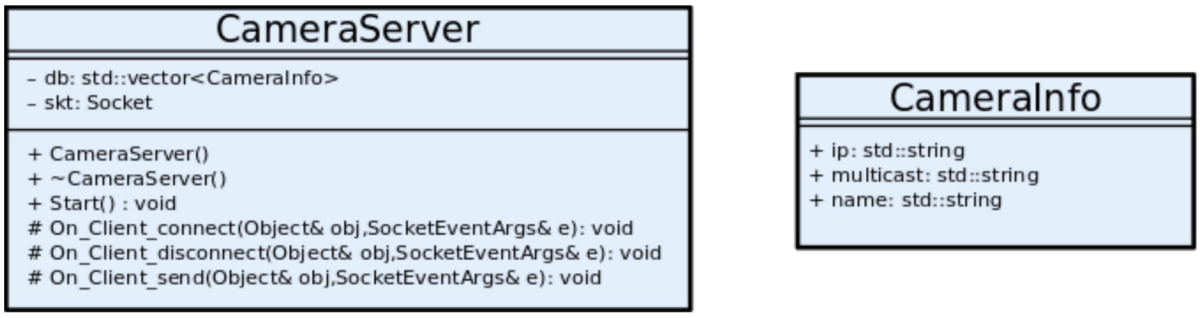


Obrázek 16: Zjednodušený vývojový diagram aplikace na straně serveru

Struktura *CameraInfo:*

Tato struktura slouží pro ukládání názvů a adres kamer. Jejíž instance je použitá ve třídě *CameraServer,* kde je použitá jako list (std::vector<CameraInfo>)

Struktura *CameraInfo* obsahuje:



Obrázek 17: Diagram tříd v aplikaci na straně serveru

- **std::string ip** – Adresa jednotky RPI.
- **std::string multicast** – Adresa živého vysílání.
- **std::string name** – jméno kamery.

Třída *CameraSever:*
Tato třída je hlavní třídou této aplikace, obsahuje metody pro obsluhu událostí, které jsou spouštěny připojenými klienty. Tato třída také obsahuje instanci třídy *Socket*, jejíž cílem je poskytnout otevřenou komunikaci na TCP protokolu.

Metoda *On_Client_connect* se v případě připojení klienta se zavolá, její účel je jen informativní a slouží jen pro ladicí účely, jen vypisuje adresy nově připojených klientů.

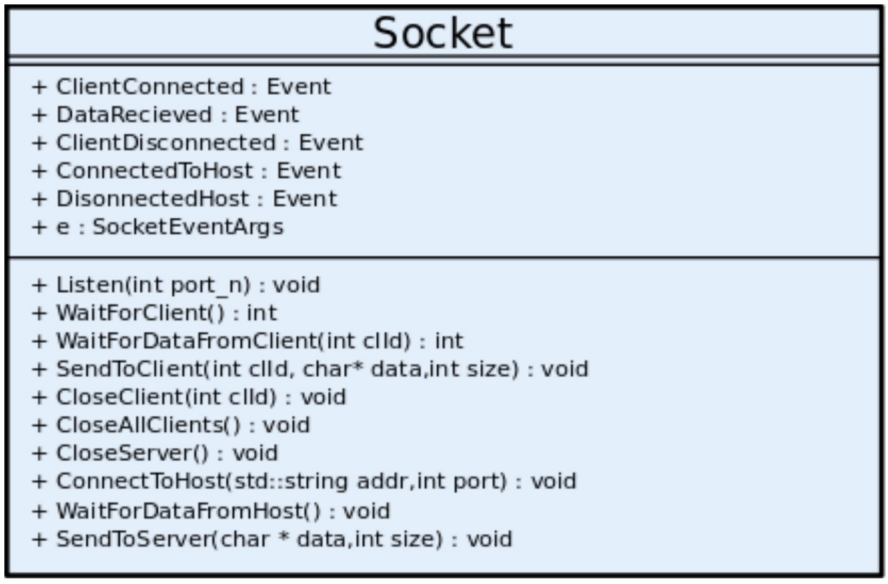
Metoda *On_Client_send* je volána vždy, když klient pošle na server žádost. Data obsahující žádost jsou do této metody předány přes její argumenty pomocí reference.

Metoda *On_Client_disconnect* je volána v případě odpojení klienta. Do této metody jsou předány argumenty, které v sobě obsahují adresu klienta, podle které se ze seznamu adres vymažou všechny záznamy, které tento klient na serveru zaregistroval.

Serverová aplikace využívá třídu *Socket*, která využívá vlastní způsob obsluhy událostí popsaný v třídě *Events*.

Třída *Socket:*
Stará se o řízení síťového TCP soketu, otevírá definovaný port a zpracovává události, které jsou propojeny s metodami, které se volají když dojde k události. Hlavními metodami této třídy jsou *Connect* a *Listen*.

Metoda *Listen* se používá pro otevření portu na kterém server naslouchá žádostem klientů. Tato metoda se používá pro server. Její parametr je číslo portu na kterém se má naslouchat.



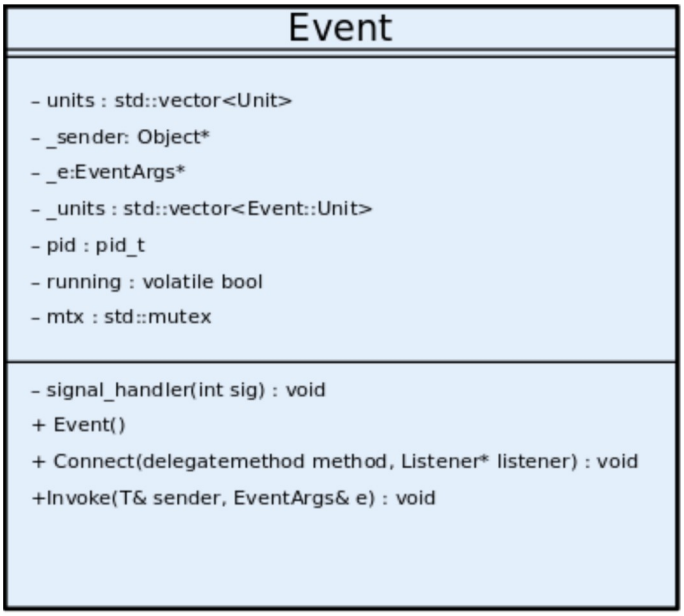
Obrázek 18: Diagram třídy Socket

Metoda *Connect* se používá na klientské straně, při použití se program chová jako klient a může na server odesílat požadavky. Jejími parametry jsou adresa serveru a port.

Třída *Event:*
Obsahuje metody, které řeší propojení mezi událostmi. Pokud událost nastane je automaticky volána metoda jenž je k události přiřazená.

Metoda *Connect* se používá pro propojení události s metodou. Její parametry jsou: ukazatel třídy v níž se nachází metoda a adresa metody.

Metoda *Invoke* se používá pro vyvolání události. Při vyvolání události se zavolá k ní přiřazená metoda, která je vždy volána v hlavním vlákně programu i když je *Invoke* voláno z jiného vlákna. Třída je uzpůsobena aby zdávala zpracovat i více událostí najednou – využívá frontu událostí.



Obrázek 19: Diagram třídy Event

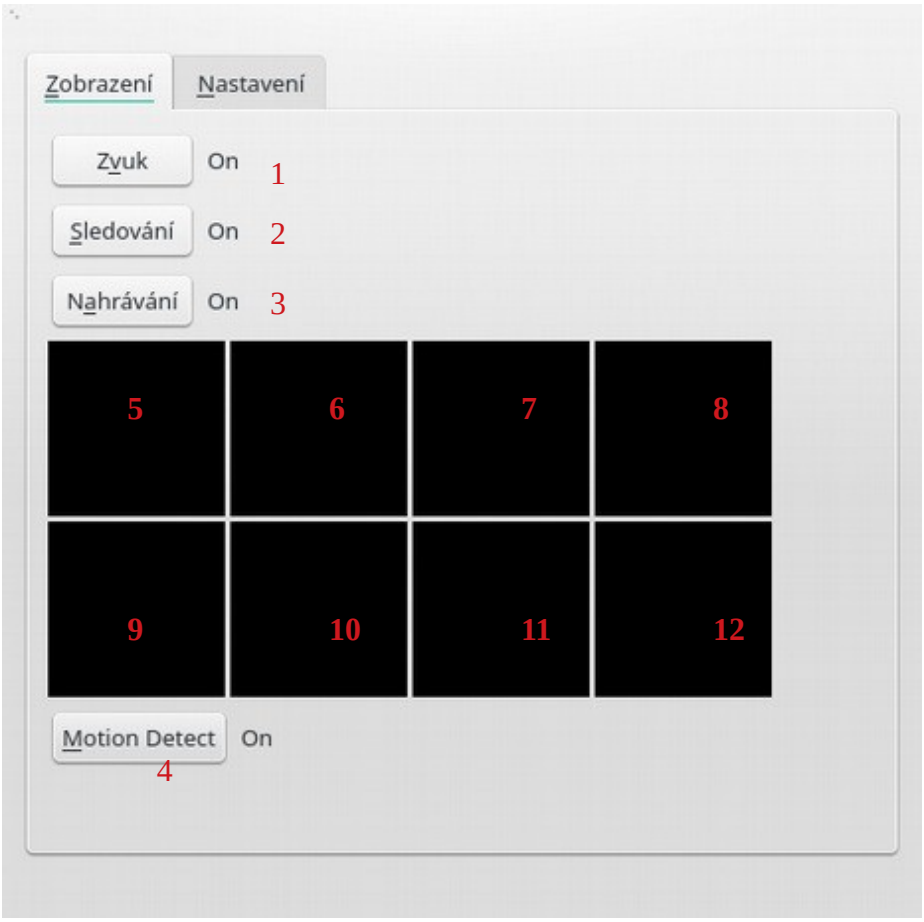
5.4 Realizace software na straně multimediálních jednotek

Software používaný v multimediálních jednotkách je napsaný v C++ s použitím frameworku Qt. Framework Qt je zahrnuje grafické knihovny pro tvorbu oken a jejich komponentů. Grafické rozhraní již bylo popsáno v návrhu viz obrázky 20 a 21.

5.4.1 Realizace grafického rozhraní

Software je navržený s grafickým rozhraním, které je rozděleno na dvě části se kterými může uživatel pracovat. V první části jsou zobrazené miniatury kamer, které si může uživatel kliknutím zvětšit. Popřípadě se miniatura zvětší sama když je na jejím obraze zachycen pohyb. Dalším nastavením je zapnutí přehrávání zvuku a sledování videa a nahrávání videa ze svých připojených kamer. Také je zde možnost zapnout nebo vypnout detekce pohybu.

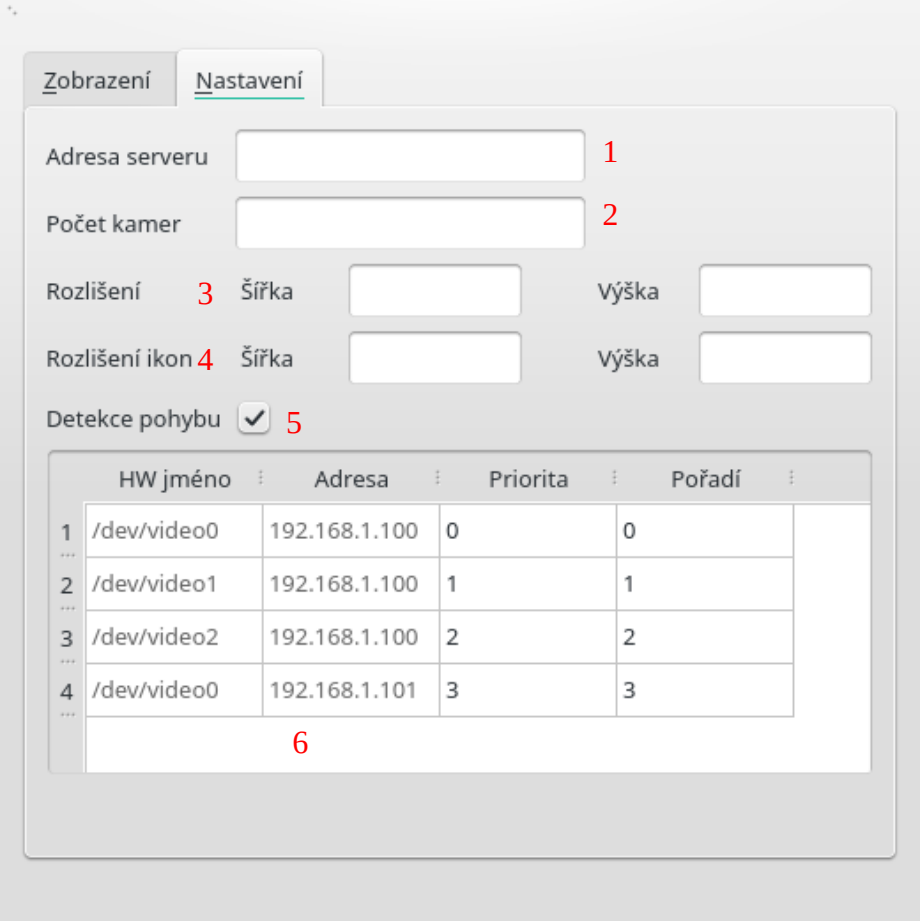
Druhá část obsahuje nastavení, kde uživatel zvolí adresu serveru se seznamem kamerových adres. Uživatel si zvolí maximální počet kamer, které chce zobrazit. Uživatel si zvolí rozlišení videa v plné velikosti a rozlišení miniatur. Dále je zde opět možnost vypnout nebo zapnout detekci pohybu. V dolní polovině okna se nachází seznam kamer nalezených v síti. U něj lze nastavit názvy kamer a jejich pořadí v jakém se budou zobrazovat v první části kde jsou zobrazeny miniatury. Uživatel si může určit prioritu detekce pohybu u těchto kamer.
Priorita detekce pohybu znamená, že když je pohyb zachycen více kamerami pak se automaticky zvětší obraz té kamery, která má vyšší prioritu.



Obrázek 20: Grafické rozhraní, část první

Seznam prvků grafického rozhraní (část č. 1):

Číslo prvku	Význam
1	Zapnutí přehrávání zvuku
2	Tlačítko zapnutí/vypnutí sledování kamer
3	Zapnutí nahrávání a vysílání obrazu z vlastních kamer
4	Zapnutí detekce pohybu
5 - 12	



Obrázek 21: Grafické rozhraní, část druhá

Seznam prvků grafického rozhraní (část č. 2):

Číslo prvku	Význam
1	Vstupní pole – adresa serveru
2	Vstupní pole – počet kamer
3	–||– rozlišení zvětšeného obrazu
4	–||– rozlišení miniatur
5	Zapnutí detekce pohybu
6	Seznam dostupných kamer a jejich adres a uživatelským nastavením pořadí a priority detekce pohybu

5.4.2 Vysílání obrazu z kamery připojené k multimediální jednotce

Pro vysílání obrazu je použit program FFMPEG. Ten video nejdříve zaznamená jako proud, který poté zkomprimuje a odešle na danou adresu. Ke spuštění tohoto programu dojde jakmile uživatel povolí nahrávání (viz obrázek 20 tlačítko nahrávaní).

Příklad použití programu FFMPEG:

```
ffmpeg -f v4l2 -i /dev/video0 -vcodec h264 -preset ultrafast -tune zerolatency -b:v 1000k  -f mpegts udp://224.0.0.1:8080
```

- **-f v4l2** – použití ovladače Video4Linux2 (ovladač zachytávajících video zařízení).
- **-i /dev/video0 –** cesta ke kamerovému zařízení.
- **-vcodec h264 –** výběr kodéru H264
- **-preset ultrafast –** nastavení účinnosti kodéru, ultrafast je rychlý ale méně účinný, opakem je veryslow, který je sice velmi pomalý ale mnohem účinnější, pro živé vysílání ale nevhodný.
- **-tune zerolatency –** nastavení pro kodér. Při použití tohoto parametru bude použito větší množství kompletních snímků při mezi snímkové- kompresi. Častější vysílání kompletních snímků vede ke snížení latence.
- **-b:v 1000k –** nastavení datového toku videa 1000kb/s.
- **-f mpegts –** výstupní formát mpegts
- **udp://224.0.0.1:8080 –** vysílat na adrese 224.0.0.1 s použitím UDP na portu 8080.

FFMPEG běží jako proces na pozadí a je volán pomocí funkce *popen().* Což je funkce v jazyku C, která umožňuje spouštět proces a získat z něho výstup. Jméno programu však není přímo předáváno do funkce *popen(),* ale je předáváno jméno programu a jeho parametry za nimiž následuje *& echo $!*. Za parametrama následuje symbol & ten proces spustí na pozadí, *echo $!* vypíše PID. PID je identifikační číslo procesu, pomocí něj se dá s procesem dále pracovat (například ukončit jej apod.).

Například:
ffmpeg -f v4l2 -i /dev/video0 …. & echo $!

Pro manipulaci s procesy byla vytvořená třída *ExternalProcess.*

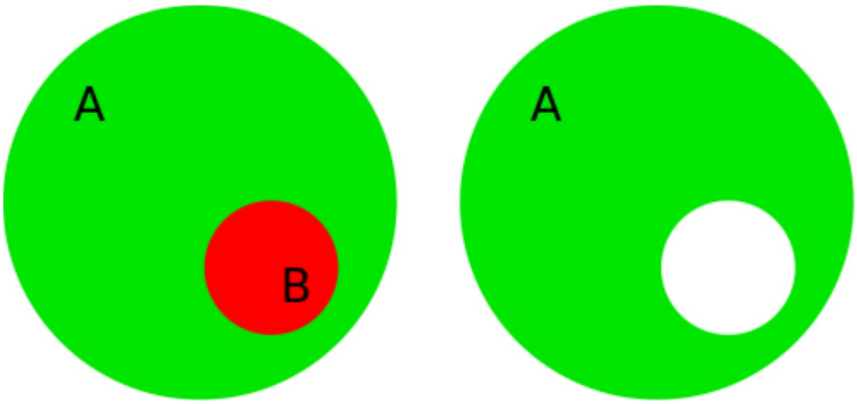
```
class ExternalProcess {
  protected:
  unsigned int pid;  //PID spuštěného procesu
```

```
    std::string path; //název programu a jeho parametry se kterými má být spouštěn

    public:
    void Start(); //Spuštění procesu
    void Stop(); //Ukončení procesu
    ExternalProcess(std::string path); //Konstruktor, do parametru je předaný název programu s
paramtry
};
```

5.4.3 Získání volné adresy pro vysílání

Multimediální jednotka musí vždy vysílat na volné skupinové adrese, nelze vysílat na stejné adrese a portu kde již vysílá jiná jednotka.

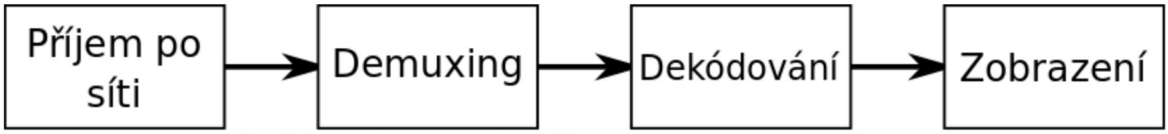
Pro získání volné adresy musí multimediální jednotka požádat server o seznam již obsazených adres. Volnou adresu získá tak že poslední adresu inkrementuje tak aby se nacházela v intervalu 224.0.0.0 – 239.255.255.255



Obrázek 22: Množina A – celkový rozsah skupinových adres, B – množina použitých adres. Vpravo po odečtení A \ B

5.4.4 Příjem obrazového signálu po síti

Multimediální jednotka, která zná vysílací adresy jednotlivých výběru a má zadáno uživatelem, které adresy má používat pro sledování videa se připojí na tyto adresy s použitím knihovny *libvlc*. Tato knihovna používá dekodéry různých formátů a funkce na příjem multimédii po síti. Automaticky umí dekódovat a demuxovat video a je schopná renderovat video uvnitř formulářového prvnku okna.



Obrázek 23: Blokové schéma příjmu multimédia po síti

Příjem videa po síti probíhá v několika krocích:

- **Příjem po síti –** Data která nesou informace o videu se nejdříve přijmou prostřednictvím UDP portu na dané skupinové adrese. Tato data jsou zakomponované do multimediálního kontejneru.

- **Demuxing –** Data přijatá po síti obsahují informace o formátu videa i audio stopě. Aby se tyto dvě stopy rozlišili musí se provést demuxing, jehož výstupem jsou oddělené stopy a informace o jejich formátech.

- **Dekódování –** Jednotlivé stopy jsou zakódované ve svých formátech (například H264). Dekódování provádí dekodér a převádí komprimovaná ze kterých lze získat jednotlivé snímky a PCM hodnoty. Video má po dekódování formu jako matice pixelů.

- **Zobrazení –** Video je zobrazeno na zobrazovacím prvku v okně. Každý snímek je vykreslován pixel po pixelu na zobrazovacím prvku v intervalech v jakém přicházejí jednotlivé snímky.

Knihovna *libvlc* je použitá pro zobrazování živého vysílání videa, k detekci pohybu je použitá knihovna *OpenCV.*

Knihovna *libvlc*, má mnoho funkci pro nastavení přehrávání, pro zjednodušení její používání byla vytvořená struktůra, která postup nastavení zjednodušuje. Tato strukture se nazývá VlcPlayer.
Struktura VlcPlayer:

```
struct VlcPlayer {
  libvlc_instance_t *_vlcinstance;
  libvlc_media_player_t *_mp;
  libvlc_media_t *_m;
  VlcPlayer() {
    const char * const vlc_args[] = {"--verbose=2" };
    this->_vlcinstance = libvlc_new(sizeof(vlc_args) / sizeof(vlc_args[0]), vlc_args);      //inicializace vlc instance
  }
  void setMedia(QString url) {
    _m = libvlc_media_new_location(_vlcinstance,url.toStdString().c_str());      //inicializace média s URL
    _mp = libvlc_media_player_new_from_media(_m);              //inicializace přehravače
  }
  void setVideoOutput(QWidget* wgt) {
    int windid = wgt->winId();
    libvlc_media_player_set_xwindow (_mp, windid ); //nastavení video výstupu na cílový widget
  }
  void play() {
      libvlc_media_player_play (_mp );
  }
};
```

Příklad zachytávání živého vysílání v kódu C++ s využitím OpenCV:

```
VideoCapture cap;
cap.open("udp://224.0.0.1:8080/");  /*otevření pojení k živému mysílání*/
Mat mat;
while(true){
   cap >> mat;  //zachynení snímku
   mat.data //obsahuje matici pixelů
   if (waitKey(30) >= 0)
      break;
}
```

5.4.5 Zobrazení přijatých video dat

Zobrazení videa v Qt je zprostředkováno pomocí formulářového prvku *QLabel*. Tento prvek je sice primárně určen pro textové nápisy, ale lze jemu přiřadit i obrazová data. Pro zjednodušení operace se zobrazováním obrázků byla vytvořená třída *QImageWidget*, která dědí od třídy *QLabel*, a umožňuje také odchytávání události kliknutí.

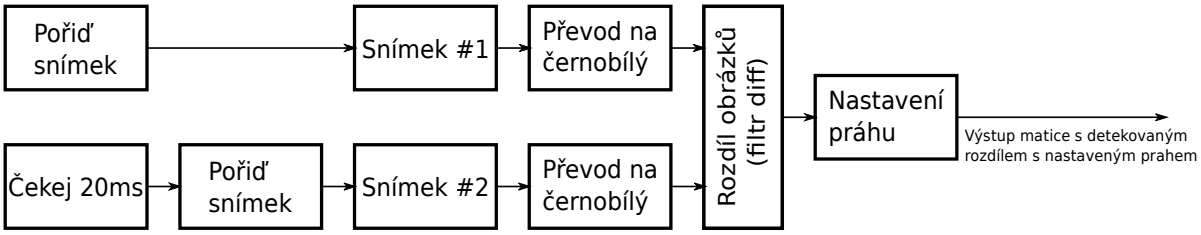
Třída *QImageWidget*, obsahuje tyto vlastnosti:

```
 QImageWidget(int w, int h); // konstruktor s rozlišením obrázku
   virtual ~QImageWidget(); //destruktor
   void setPixel(int x, int y, uint color); // nastavení hodnoty pixelu pomocí reprezentace barvy v unsigned int
   void setPixel(int x, int y, const QColor & color);// totéž co výše ale barva je uvedena v QColor
   void setCVMat(uchar* mat, unsigned long size,unsigned int bpp); // kompletní nastavení matice barev obrázku
   void Redraw(); // aktualzace vykreslení obrázku
   void resizeable(bool r);
   explicit QImageWidget(QWidget* parent = Q_NULLPTR, Qt::WindowFlags f = Qt::WindowFlags()); // přiřazení funkčnosti odchytávání události

   signals:
      void clicked(QImageWidget &img); //signál kliknutí

protected:
   void mousePressEvent(QMouseEvent* event); //pomocná metoda která reaguje na kliknutí na QLabel a spustí signál clicked
   QImage * image;
```

5.4.6 Detekce pohybu na přijatých videích

U sledovaných video přenosů si uživatel může vybrat zda chce povolit detekci pohybu. Detekce pohybu spočívá v porovnávání jednotlivých snímků. Porovnávání snímků probíhá způsobem při

kterým se porovnávají jednotlivé pixely. Tento způsob může být výkonově náročný, ale nároky lze snížit převedením snímků na černobílé a následným rozdílem je porovnat. Tyto dva snímky, které se mají porovnat jsou pořízeny s určitým časovým odstupem, například 20 ms, nebo každý desátý snímek. Dále jsou převedeny na černobílé a následně jsou zpracované filtrem *absdiff,* který spočte rozdíl mezi jednotlivými pixely a vytvoří rozdílový snímek. Tento snímek je pak zpracován filtrem *threshold*, čímž se dá natavit práh detekce rozdílu, jimž lze nastavit toleranci pro dekekci pohybu a ignorovat tak šum.



Obrázek 24: Blokové schéma převodu snímků na černobílý formát a jejich porovnání

Výsledkem je poté prahový snímek, který obsahuje pouze bílé nebo černé pixely, a nic mezi tím. Tento prahový snímek oproti barevným a černobílým snímkům zjednodušuje a zrychluje práci s detekci pohybu. Míra pohybu ve snímku lze zjistit podle počtu bílých pixelů.

Obrázek 25: Prahový snímek ruky v pohybu

K rozlišení míry pohybu v obraze se používá algoritmus, který z výstupního prahového snímku sečte všechny pixely jejichž hodnota je nenulová. Tím se získá celkový počet pixelů na nichž byl zaznamenám pohyb.

Kód detekce pohybu:

```
while (capturing) {
  if (cap.isOpened())
     cap >> mat1;  //zachycení snímku.
  else break
  if (cap.isOpened())
      cap >> mat2; //zachycení druhého snímku
  else break
  cv::cvtColor(mat1,gray1,cv::COLOR_BGR2GRAY);          //převod na černobílý
  cv::cvtColor(mat2,gray2,cv::COLOR_BGR2GRAY);
  cv::absdiff(gray1,gray2,diff);          //rozdíl dvou snímků
  cv::threshold(diff,thrs,50,255,cv::THRESH_BINARY);          // threshold snímku
  double mval=0;
  for ( int i=0 ;i< thrs.cols*thrs.rows;i++ ) {
     mval += thrs.data[i];          //sečtení všech pixelů
  }
  mval = mval/255.00;          //přepočet : získání počtu pixelů s pohybem
  if (mval > 500 ) {
     motion.Invoke(*this,ev);          // vyvolání události detekovaného pohybu
  }
}
```

Následně je poté volána metoda *MotionDetected.* Tato metoda příjímá parametr typu *MotionEventArgs*, což je třída obsahující multicastovou adresu a prioritu snímku. Priorita se porovná s předchozí prioritou naposledy zjištěného snímku a vybere se číslo s nižší hodnotou. Následně se spustí přehrávání videa na ve velkém okně.

6. Verifikace a testování

Testování proběhlo na třech jednotkách RPi 3. Bylo ověřeno, že systém je schopen automaticky nabídnout volné skupinové adresy každému vysílacímu zařízeni s kamerami.

Cílem této verifikace je zjistit zpoždění mezi přehrávaným videem a nahrávanou scénou a to nejen zpoždění při přehrávání, ale i prodlevu při spuštění přehrávání vzniklou během inicializace spojení mezi přehrávačem a vysílacím uzlem.

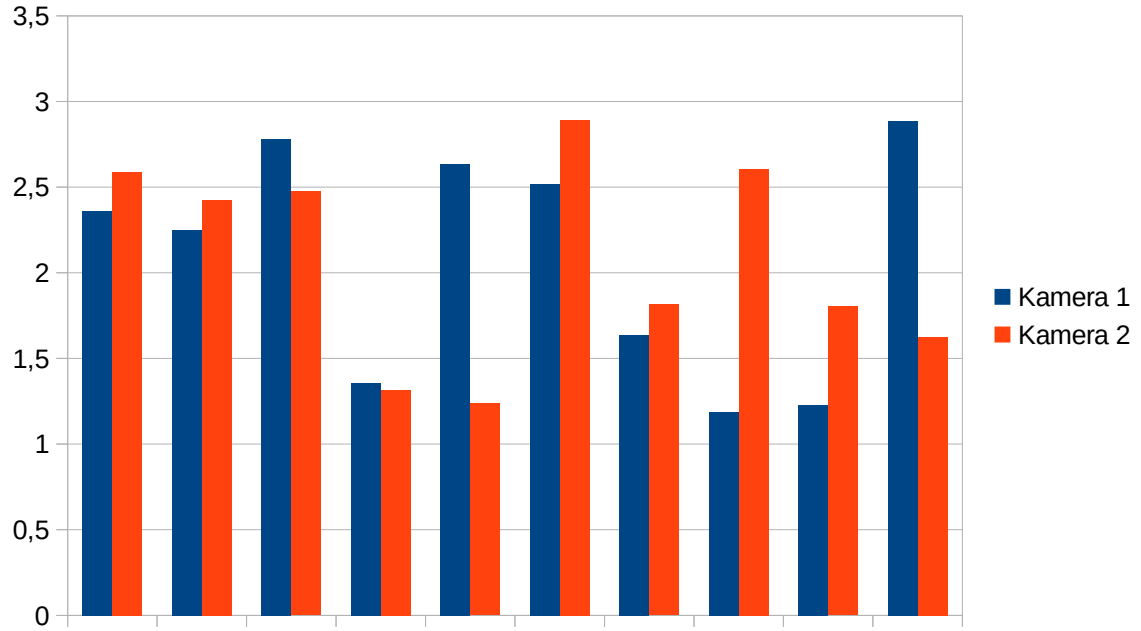
6.1 Verifikace detekce pohybu

Verifikace detekce pohybu byla prováděná pomocí tří Raspberry Pi 3 B+, dva z nich vysílaly a jeden přijímal. Software na této straně byl schopen při zachyceném pohybu na zvoleném videu automaticky otevřít okno se zvětšeným formátem tohoto videa. Software je schopen během několika sekund otevřít zvětšené okno, toto zpoždění je způsobeno spouštěním další instance v knihovně libVLC, která se pokouší připojit na druhý video stream, který obsahuje video s vyšším rozlišením. Okno se otevře s videem až instance libVLC dostane dostatečně velký objem dat, které uloží do bufferu a následně je začne dekódovat.

Jednotlivé naměřené časy byly časy skriptem, který změnil pozadí obrazovky snímané kamerou na bílou a poté spustil měření času do doby než se otevře okno

Tabulka 8: Naměřená zpoždění detekci pohybu ze dvou kamer (údaje jsou v sekundách)

Měření	Kamera 1	Kamera 2
1	2,359	2,588
2	2,251	2,423
3	2,781	2,474
4	1,358	1,316
5	2,637	1,237
6	2,517	2,890
7	1,638	1,815
8	1,185	2,606
9	1,226	1,803
10	2,889	1,626



Obrázek 26: Graf zpoždění při živém vysílání

Průměrné zpoždění u obou kamer je zhruba 2,08 sekund.

6.2 Optimalizace citlivosti detekce pohybu

Citlivost vyhodnocování pohybu lze nastavit vhodnou volbou podmínky if (mval > 1000). Proměnná mval v obsahuje číselný údaj, který je roven počtu pixelů na kterých byl zaznamenaný rozdíl mezi dvěma porovnanými snímky. Pokud tuto podmínku snížíme (např. if (mval > 500)), zvýšíme tím citlivost detekce pohybu. Citlivost lze také změnit úpravou parametru určující práh detekce rozdílů mezi snímky

```
cv::threshold(diff,thrs,THRESHOLD,255,cv::THRESH_BINARY);
```

Změnou parametru THRESHOLD, respektive jeho snížením, lze opět dosáhnout vyšší citlivosti na pohyb.

6.3 Optimalizace odezvy video vysílání

Cílem optimalizace je snížit zpoždění vysílaného videa oproti skutečnosti. Zpoždění se projevuje dvěma způsoby:

1. Zpoždění pří započetí příjmu videa – přijímací proces před započetím zobrazování nejprve čeká na přijetí I-snímku, poté začne přehrávat. Pro snížení tohoto zpoždění je třeba na vysílací straně zvýšit počet I-snímků (přidání parametru -zerolatency do řádku FFMPEG).

2. Zpoždění mezi skutečností a sledovaným videem – tento druh zpoždění je způsoben na obou stranách. Na sledovací straně je způsobeno větším bufferem, do nějž se snaží přehrávač uložit větší objem dat, díky němu vzniká zpoždění. Toto zpoždění lze snížit nastavením parametru u libvlc: `--network-caching=500`

Po těchto optimalizacích zpoždění videa dosáhlo průměrně 1,5 sekund.

7. Závěr

Cílem této bakalářské práce bylo realizovat síťový kamerový systém, který je schopen vysílat větší množství kamer v síti a zároveň sledovat jejich přenosy na stejných zařízeních, které vysílají.

V rámci návrhu jsou popsány funkční principy a algoritmy jednotlivých částí. Je zde navržen systém komunikace mezi jednotlivými multimediálními jednotkami, kde pomocí serveru s databází adres a jmény je zprostředkována možnost vyhledávání dostupných kamer.

V realizaci byly podrobněji rozepsány algoritmy a jejich ukázky zdrojových kódů. Bylo zde popsáno grafické rozhraní, jeho jednotlivé části a popis funkcí formulářových prvků.

V rámci testování byly použity tři vestavěné jednotky Raspberry Pi 3. Test zpoždění byl proveden se dvěma jednotkami Raspberry jenž každá z nich měla jednu připojenou kameru. Video přenosy z těchto kamer byly vysílány do třetího Raspberry u nějž se měřilo zpoždění vysílaného videa. Celkové zpoždění u obou kamer (měřeno zvlášť) bylo přibližně 2 sekundy. Následně byla provedena optimalizace, po níž došlo k mírnému snížení zpoždění cca o 0,5 sekundy.

Snaha při optimalizaci byla zmenšit zpoždění při přehrávání živého vysílání. Tato zpoždění byla před započetím přehrávání, kdy se naplňoval buffer, a další zpoždění mezi reálnou scénou a přehrávaným videem. V textu byly popsány příčiny jakými tato zpoždění vznikají. Na základě těchto příčin se provedla optimalizace. Po všech optimalizacích bylo zpoždění menší než 2 sekundy menšímu zpoždění se již nepodařilo dosáhnout.

Z uživatelského pohledu tento systém funguje jako sledovací zařízení s možností sledovat více míst najednou, ale také jako dorozumívací zařízení mezi mnoha lidmi najednou s podporou zvuku.

Použitá literatura

[1] Mallat, S. *A Wavelet Tour of Signal Processing : The Sparse Way. With contri-butions from Gabriel Peyré.Academic Press*, třetí vydání, 2009, ISBN 978-0-12-374370-1.

[2] Richardson, I. E. G. *H.264 and MPEG-4 Video Compression: Video Coding forNext-generation Multimedia.* Wiley, 2003, ISBN 978-0-470-84837-1.

[3] Salomon, D. *Data Compression: The Complete Reference.* Springer-Verlag, Čtvrtévydání, 2007, ISBN 978-1-84628-602-5.

[4] Taubman, D. S. *Marcellin, M. W.:JPEG2000: Image Compression Fundamentals,Standards and Practice.* Springer, 2002, ISBN 978-1-4613-5245-7.

[5] OZER, Jan. *Producing Streaming Video for Multiple Screen Delivery.* Doceo Publishing, 2013. ISBN 978-0976259541.

[6] AUSTERBERRY, David. *The Technology of Video and Audio Streaming.* 2 edition. Focal Press, 2004. ISBN 978-0240805801.

[7] GHANBARI, Mohamed. *Standard Codecs: Image Compression to Advanced Video Coding.* London,UK: The Institution of Engineering and Technology, 2003. ISBN 978-0852967102.

Seznam příloh

Součásti bakalářské práce je CD.

Seznam příloh na CD

Zdrojové kódy aplikací zabalených v archívu

Návod na kompilaci zdrojových kódů